Thomas Becker
02.02.1983 in Bonn
Matriculation number: 517248
thomas.heinrich.becker@web.de
FH Düsseldorf, University of Applied Sciences

Fiducial recognition with infrared multi-touch frames

Thesis to obtain the degree Bachelor of Media Engineering

Supervisor:

Professor Dr. Eng. Univ. of Tsukuba Jens Herder
FH Düsseldorf, University of Applied Sciences
Department of Media

Dr. Rolf Landua
Head of Physics Department - Education Group, CERN
Organisation Européenne pour la Recherche Nucléaire

August 2012

Abstract

The recent introduction of multi-touch sensitive displays has brought with it the challenge of recognizing tangibles on these kinds of screens.

There a several wide spread and/or sophisticated solutions to fulfil this need but they seem to have some flaws. Printed patterns, for example, can be recognized quite well by displays with integrated optical sensors, but those systems are either time-consuming to calibrate or don't work properly in case of extensive use of illumination.

One popular system at the time of writing is an overlay frame that can be placed on a normal display with the corresponding size. The frame creates a grid with infrared light emitting diodes. The disruption of this grid can be detected and messages with the positions are sent via USB to a connected computer. This system is quite robust in matters of ambient light insensitivity and also fast to calibrate. Unfortunately it is not created with the recognition of tangibles in mind and printed patterns can not be resolved.

This thesis is an attempt the create fiducials that are recognized by an infrared multi-touch frame as fingers. Those false fingers are checked by a software for known patterns. Once a known pattern (= fiducial) has been recognized its position and orientation are send with the finger positions towards the interactive software.

The usability is tested with an example application where tangibles and finger touches are used in combination.

One key aspect is the use of the system by multiple interactors on a 55 inch screen.

Keywords: low cost multi-touch infrared overlay frame fiducial tangible recognition

Abstract in German

Mit zunehmender Nutzung von Multitouchdisplays stieg auch das Verlangen Objekte auf diesen berührungsempfindlichen Displays zu erkennen. Hierfür gibt es einige Lösungen die gut funktionieren, aber systemabhängig sind. Gedruckte Muster, zum Beispiel, lassen sich bei Displays mit integrierten optischen Sensoren oft sehr gut erkennen, aber diese Systeme sind entweder aufwendig zu kalibrieren oder aber reagieren empfindlich auf Umgebungslicht.

Ein zur Zeit gängiges System ist ein Rahmen der auf ein Display aufgesetzt wird. In diesem Rahmen befinden sich Infrarotleuchtdioden und Empfänger, welche ein Gitter aus Infrarotstrahlen erzeugen und bei Unterbrechung die Position des Fingers als Signal an den angeschlossenen Computer weitergeben. Dieses System ist sehr robust in Bezug auf äußere Lichteinflüsse und benötigt minimalen Kalibrieraufwand.Leider können die gedruckten Muster mit diesem System nicht erkannt werden.

Diese Arbeit beschreibt den Versuch “fiducials” zu erzeugen welche von dem Infrarotrahmen als Finger wahrgenommen werden. Diese “falschen Finger” werden dann in ihrer Konstellation zueinander auf bekannte Muster überprüft. Diese Überprüfung findet durch eine Software statt, welche die Nachrichten mit den Fingerpositionen abfängt und falls ein Muster erkannt wird, die Position und Orientierung des erkannten fiducials in den Nachrichtenstrom des Rahmens einfügt und weitersendet.

Untersucht wird die Anwendungstauglichkeit mit Hilfe einer Beispielanwendung in der fiducials und Berührungen in Kombination genutzt werden. Ein wichtiger Aspekt ist die Nutzbarkeit bei Interaktion mehrerer Personen auf einem 55 Zoll großen Bildschirm.

Stichworte: low cost multi-touch infrarot Rahmen fiducial tangible Erkennung

Acknowledgment

I would like to thank Professor Herder for his great support throughout my whole 15 months at CERN. His patience and interest in my work have been a huge benefit for me.

I have to thank Rolf Landua for the support of this thesis, my project and most of all for the time in Geneva. The opportunity to experience such an amazing place with so many kind and interesting people definitely gave my life some spin. I really enjoyed being a part of the PH-EDU group.

And of course I will forever be thankful for the patience of my colleague João Cordovil Bárcia, and all the other members of the CERN MediaLab, for enduring me screaming like a monkey.

THANK YOU

Table of contents

1. Introduction

The recent introduction of multi-touch sensitive displays has brought with it the challenge of recognizing tangibles on these kinds of screens.

There a several wide spread and/or sophisticated solutions to fulfil this need, but they seem to have some flaws. Printed patterns, for example, can be recognized quite well by displays with integrated optical sensors, but those systems are either time-consuming to calibrate and / or don't work properly in case of extensive use of illumination.

One popular system at the time of writing is an overlay frame that can be placed on a normal display with the corresponding size. The frame creates a grid with infrared light emitting diodes. The disruption of this grid can be detected and messages with the positions are sent via USB to a connected computer. This system is quite robust in matters of ambient light insensitivity and also fast to calibrate. Unfortunately it is not created with the recognition of tangibles in mind and printed patterns can not be resolved.

This thesis is an attempt the create fiducials that are recognized by an infrared multi-touch frame as fingers. Those false fingers are checked by a software for known patterns. Once a known pattern (= fiducial) has been recognized its position and orientation are send with the finger positions towards the interactive software.

The usability is tested with an example application where tangibles and finger touches are used in combination.

1.1. Who can profit from this thesis?

This thesis can help creators of interactive soft- and hardware to extend the capabilities of their infrared overlay with fiducial recognition.

Low cost approach

Once the frame has been purchased the additional costs for the tangibles are minimal.

Tangibles can be cut out of Makrolon, Plexiglas, Teflon or other materials. Simple prototypes made out of Compact Discs and wine cork are also possible and reduce the additional costs to a minimum.

No sophisticated knowledge is needed to create the tangibles. They are totally passive, no electronic components are needed.

1.2. Open Source Code

The complete code under the GPLv3 can be downloaded from github:

https://github.com/hoshijirushi/Vega

1.3. Status quo

There are several approaches to recognize tangibles on multi-touch displays. Often they have some of those flaws:

- Bulky
- Difficult to build
- Only working if the ambient light is setup in a specific way
- Need calibration often and / or extensive

But most of them work quite reliable in terms of fiducial recognition.

The reactable like approach

One of the most used techniques for fiducial or marker recognition is the optical tracking of printed markers with an infrared camera.

The basic setup for this system is a surface with back projection. On the same side as the projector, the back, is an infrared camera positioned. It records an infrared image of the projection surface. With this technique it is possible to recognise touches and printed markers on the “display surface” which then can be identified by pattern recognition techniques. [Kaltenbrunner, M. & Bencina, R. (2007)]

This system works quite well, but needs to be calibrated once the projector or the camera have been displaced.

Furthermore the system has the height of a table, due to the necessity of projectors and cameras underneath.

Another issue can be the cooling of the projector, as it generates heat.

Capacitive tangibles

Capacitive multi-touch screens measure the capacity of transparent and conducting layer on top of an insulator like glass. The position is obtained by different techniques e.g. measuring the capacity from the four corners of he display. [Wikipedia Touchscreen]

The tangible recognition with capacitive multi-touch screens is robust and accurate.

Unfortunately are capacitive screens only used in small (< 20 inch) dimensions. A at least 55 inch big screen is needed most of the time when multiple people want to interact with a screen. According to this, those screens are not suitable.

Microsoft PixelSense (former Surface)

This system is a LCD display where each pixel has RGB values plus a sensor to detect infrared light. The infrared light is emitted with the normal backlight from behind the LCD. Once an object is placed on the surface of the display, it reflects the infrared light back to the sensors in the LCD. This makes it possible to see the surface like a black and white scanner. [Microsoft PixelSense]

The Microsoft PixelSense system is at the moment of writing this document only available in a 40 inch version. Also recent tests in the UAS Düsseldorf showed that the use of this table can be difficult for example when it is used

in a studio environment. The sensors are then “blinded” by the ambient light.

Infrared touchscreen

Overlays for displays and projection surfaces can be used to extend a stock display or TV with touchscreen capabilities. Due to their frame-like design they can be easily affixed to a screen without losing image quality. In this way 3D TVs can easily be retrofitted to be touchsensitive.

Technique behind this system is a horizontal and vertical grid of infrared rays that are created by irLEDs and received by infrared light sensitive sensors. Disrupting the grid creates a signal with the position of the interference.

The system needs just a calibration to setup it’s position in relation to the screen. This is done by four touches with a finger.

Unfortunately this system detects touches in an layer above and parallel to the screen, thus rendering it unable to detect, for example, printed tangibles.

1.4. False fingers - another approach

The basic idea is to use tangibles that glide on “false fingers” on the display surface. In that manner the frame can detect those “false fingers” and provide the information on their positions to the receiving software.

The software can then determine whether the constellation of points (false fingers) is stored in a database and the object thereby is recognized.

If it is recognized the orientation and the position of the marker can be calculated.

The advantages of this system are:

- The tangibles can be directly placed on a screen.
- No projector or camera is needed
- Just a one-time calibration is needed. The maximum calibration error is bound to the ability of the frame to shift relatively to the display. This can be reduced to less than 1 mm with a professional fixation.
- A layer of glass can be placed on the screen for protection.

1.5. Use case

To check the usability of this approach it is tested with a software that requires touch and tangible interaction.

The proton / neutron builder

The test program is a software helping to visualize what certain particles are made of. One simplified case is the creation of neutron and a proton. They are made of quarks.

To create a particle on the screen the user has to place 3 quarks in a circle. If they are in the right combination they start to form a proton or a neutron and interact with each other and the shell becomes alive.

The test program is created in Ventuz and reacts to TUIO signals. The particles can be be moved with the tangibles and have a touch button in their middle. With additional buttons it tested if the false tangibles fingers trigger them when they are moved over them, which should idealy not the case.

2. The pattern algorithm

2.1. The idea in detail

Whale sharks and star trackers

The search for an existing algorithm to detect triangle patterns led to a project that was dedicated to the identification of whale sharks. [Arzoumanian, Holmberg & Norman 2005]

A research group studying populations of whale sharks faced the challenge of robustly distinguishing individuals of the species.A normal procedure to identify the sharks has been tagging the live animal. Due to a lack of good results, a different approach was chosen: Identification of the whales by their unique body pigmentation.

Photographs of whales were taken and position and time of those sightings noted. A database with pictures was created, but a manual comparison of the pictures and the identification of the whales were still a time consuming task.

The search for an automatic approach led them to a problem that is similar and has already been solved: star tracking. A form of celestial navigation.

One should imagine a satellite in space. The control system of this satellite needs to know where the satellite is located in space to adjust, for example, its orientation or orbit.

How does the system know where it is, except from a ground station transmitting this to the system?

The calculation of position and orientation is based on the same principles that our ancestors used to navigate on the earth surface: observation of the stars.

The task for a machine to calculate the position is a simple one, but the machine somehow needs to identify the stars around it.

To do this, the satellite is equipped with a camera and an electronic map of the surrounding stars. The camera is used to watch the stars. The electronic map is needed to compare the taken pictures with the known data.

The computer can compare what the camera "sees" with the data given by the electronic map and knows in which direction it is "looking" at the moment. Out of several measurements the position and orientation of the satellite can be calculated.

The comparison of what a camera is seeing with a stored map is exactly what was needed for the shark project: A picture of a whale was stored in a database and it's features extracted for comparison. Even the dotted pattern on the back of the whale sharks resembled images of stars.

In the following process every new whale picture was compared with the database and once a feature pattern was recognized a already known whale was identified.

The algorithm used for the pattern recognition was described by Edward J. Groth. [Edward J. Groth 1986]

Groth

Pictures of stars or pictures of dots on whale sharks, both can be simplified into two-dimensional coordinate lists. Those lists simply describe where a star (or a dot) is located, in x and y coordinates, on a picture (or map)

To describe relationships between features (stars or dots) one simply creates triangles between these features.

Afterwards the triangles are compared with a list of known triangles and checked for matches.

Unique triangles can be described in different ways:

- Three known sides
- A known angle and two known sides (if the angle is opposite to the longer side)
- One known side and two known angles

To make his algorithm insensitive towards magnification caused by lenses and so on, Groth chose to only compare two features of a triangle:

- The ratio of the longest side to the shortest side
- And the cosine of the angle between them

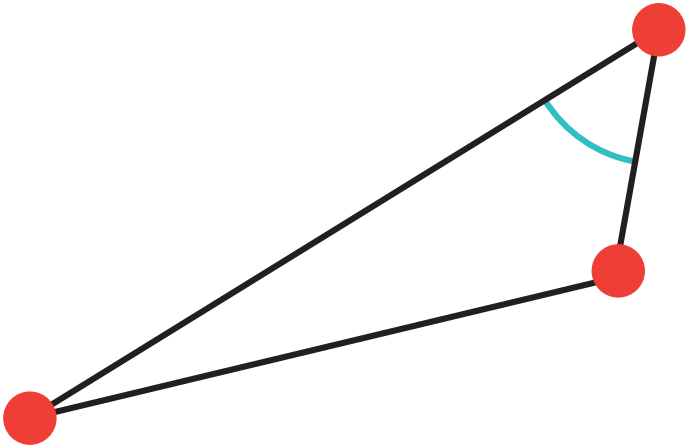


Figure 1. Three detected points with the triangle created by them. The cosine is calculated out of the angle between longest and shortest side.

Not only is this algorithm now insensitive towards magnification, it is also disregards the sense of the triangle. A mirrored triangle would still be considered as identical to the unmirrored.

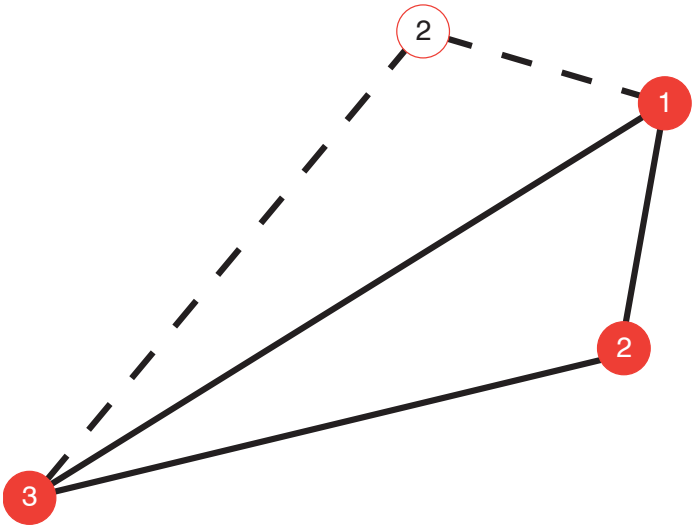


Figure 2. The solid and the dotted triangle would be considered equal by the algorithm.

The triangle ratios are compared and in case the difference between them is smaller than a user defined error rate, the triangles are considered equal concerning their ratios.

In the following step the ratio matching triangles are checked again, but this time the cosine is compared. If the error is also smaller, the triangle is a match.

If the patterns would consist of only three points this would be sufficient. However triangles are still frequently falsely recognized and one should make sure that those points that have been recognized are actually those that they

seem to be. Luckily the patterns consist of more than three points and therefore more triangles are created.

A voting system is introduced:

A matched triangle has 3 votes. It passes those votes to the points it is created from. Every pair of points gets one vote.

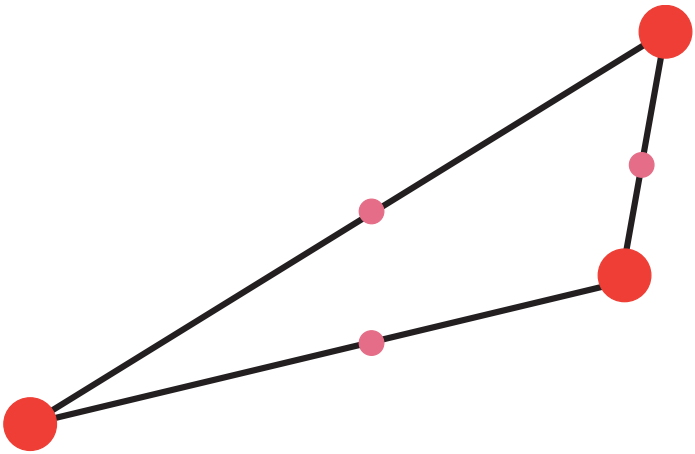


Figure 3. A triangle created by three points. Each combination of points received one vote.

The more votes a pair of points has, the more likely it is that those points are not falsely recognized.

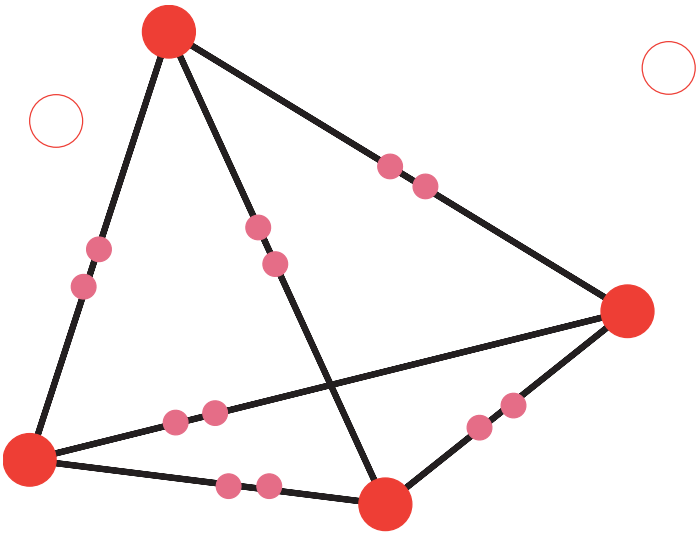


Figure 4. More created triangles result in more votes for the recognized points.

If only the features with the most votes are used for calculations, the result should be the best possible.

Disadvantages of Groth's algorithm for the tangible recognition

Groth's algorithm's tolerance towards magnification and mirroring of patterns is not required in the recognition of false fingers on an infrared overlay, in fact elevating the number of false recognitions. The corrected size and chirality of our generated fiducials enables us to optimize the algorithm slightly

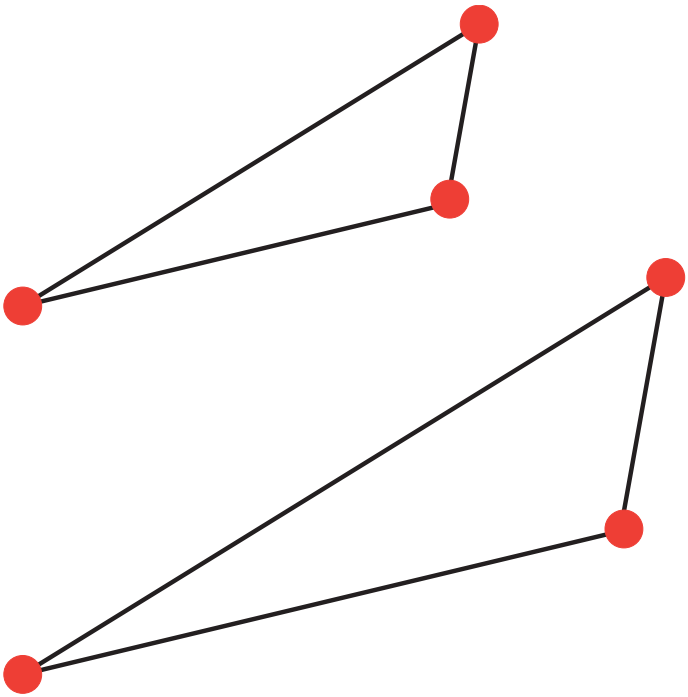


Figure 5. Both triangles appear identical towards Groth's algorithm even though they have different sizes.

The magnification doesn't change with a corrected infrared overlay and corrected tangible size neither is the tangible flipped. The algorithm can be optimized.

The altered algorithm

To simplify and therefore accelerate calculations we compare the following features of triangles:

- The length of all three sides

- The orientation of the triangle (if the alignment of certain points is defined as clockwise or counterclockwise)

The orientation is calculated by taking the longest side of the triangle. Which is v1 to v3. The shortest side is reaching from v1 to v2. If v2 is positioned on the “left” of the longest side, the triangle is declared to have clockwise orientation.

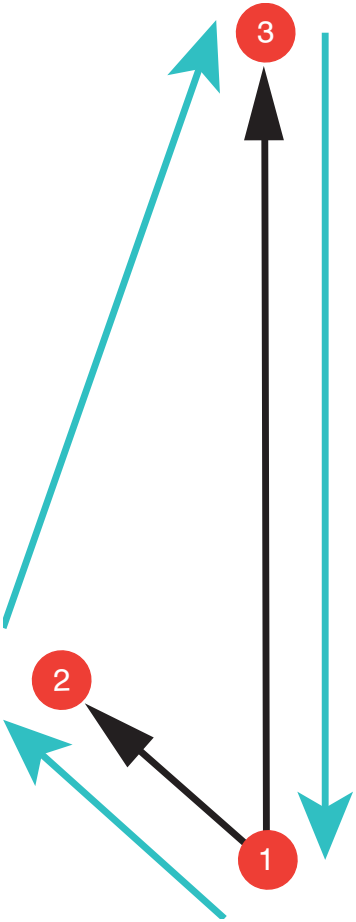


Figure 6. Triangle defined as “clockwise” oriented

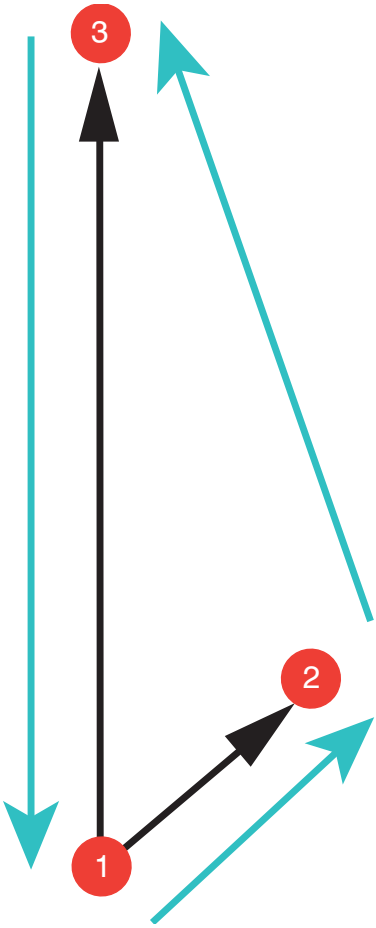


Figure 7. Triangle defined as “counterclockwise”. oriented

The condition for a clockwise orientation is:

((v3.x - v1.x)*(v2.y - v1.y) - (v3.y - v1.y)*(v2.x - v1.x)) > 0

Magnification and mirroring errors are now excluded. The recognition rate of our tangibles is improved.

2.2. Calculation of tangible position and rotation

Initial registration of a tangible

To recognize a tangible we first have to save it's features in our database.

Saved tangible points and position towards tangible center

We could just save the properties (length of the sides and orientation) of the triangles created by our tangible. With the triangles alone we could identify the tangible. But we want more, we also want to know where our tangible is and how it is orientated.

We choose a center point, e.g. the center of the infrared overlay and center the tangible above it. The position is known to us. Now we can save the horizontal and vertical distance of each feature (false finger) relatively to our center.

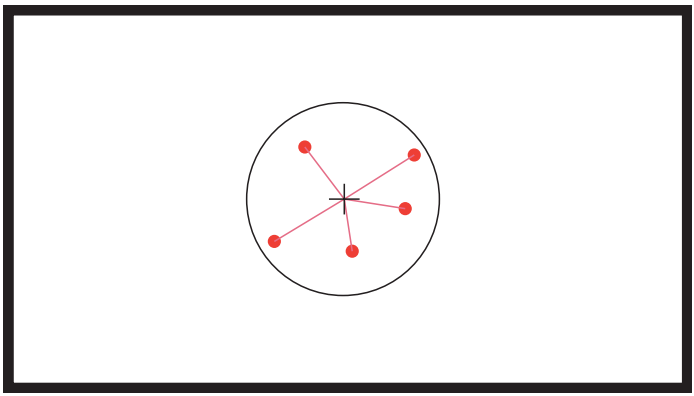


Figure 8. The tangible has been placed on the center of the overlay. The horizontal and vertical distance of all recognized points towards the center is saved.

With this relative distances it is possible to calculate the tangible center from absolute finger position values received by the screen.

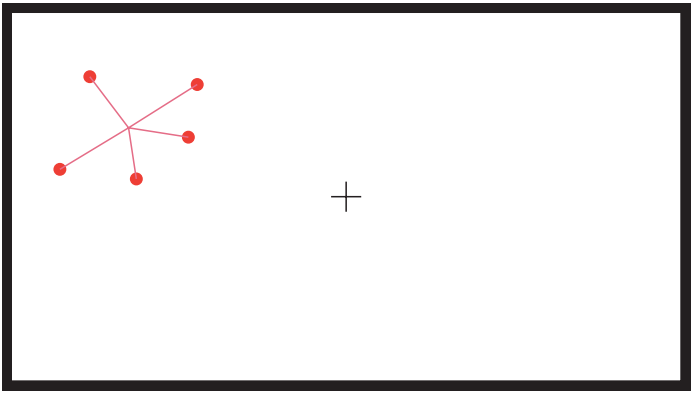


Figure 9. The momentary center of the tangible can be recreated out of the points position.

Rotation of a tangible

Once two points of the tangible are clearly identified (with the voting system), it is possible to calculate an angle between the vector spanned by those two points and a reference vector (e.g. 1,0)

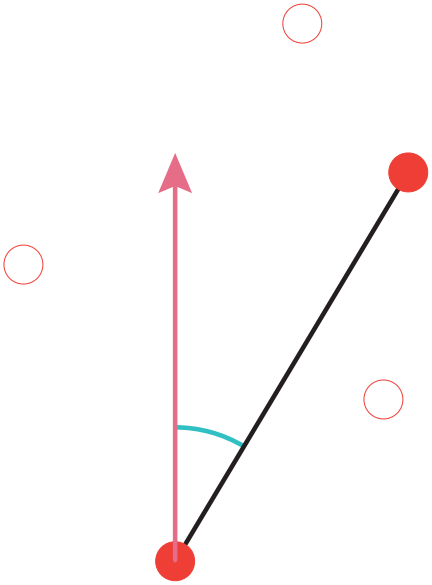


Figure 10. The angle between the reference vector (1,0) and the vector from the first point in direction of the second is calculated.

If we compare this angle with the angle those two points had towards the reference vector when they were placed for registration we can see how far the tangible has been rotated.

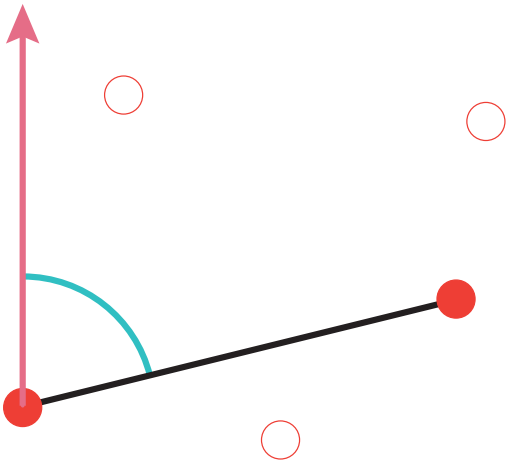


Figure 11. The angle of the points towards the reference vector when the tangible was placed in the center for calibration.

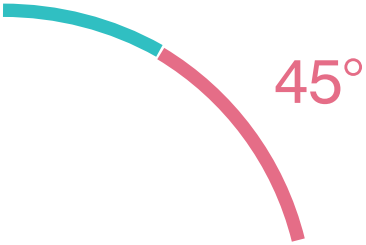


Figure 12. The tangible has been turned counter clockwise by 45 degrees.

Position of the tangible center

We take one of the clearly identified points. We saved its distance towards the center before, when we registered the tangible positioned on the registration center.

Wherever our identified point is located now on the touchscreen, the center should be in the horizontal and vertical distance we saved before.

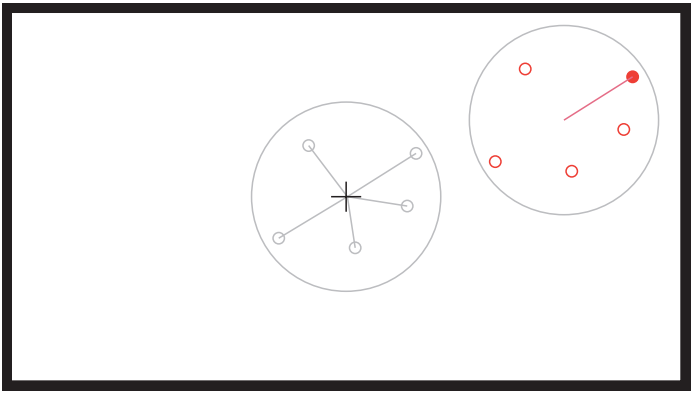


Figure 13. A vector starting from one of the cursors that have been identified, by the algorithm, to belong to the tangible. It is pointing in horizontal and vertical direction towards a spot where the center of the tangible should be now.

Figure 14. Vector leads to the tangible center.

Unfortunately the tangible might have been rotated and with this the position of the center is shifted.

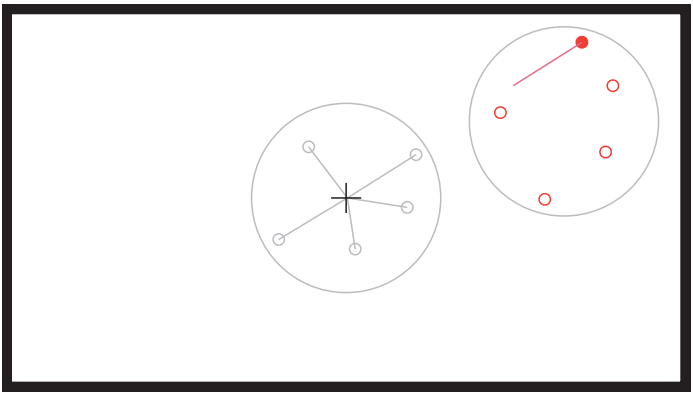


Figure 15. The original vectors from the registered points can't be used to calculate the tangible center with rotated tangibles.

By rotating the vector reaching from the identified point to the center by the value of the tangible rotation we end up with the correct center.

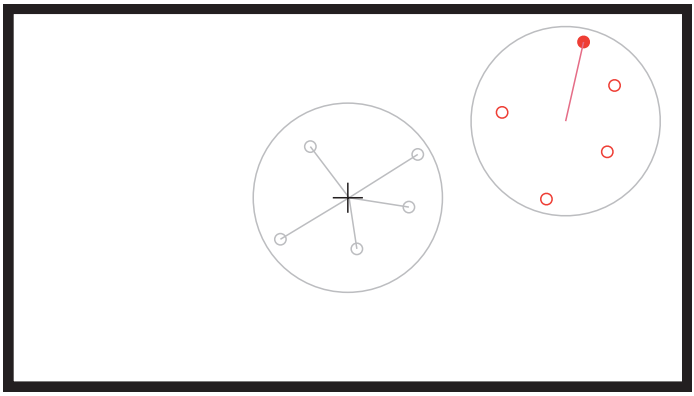


Figure 16. After rotating the registration vectors by the tangible rotation, the point again towards the tangible center.

Screen ratio correction

The position of a finger we receive from the touch frame consist of two coordinates. Each x and y value is in a range of 0 and 1.

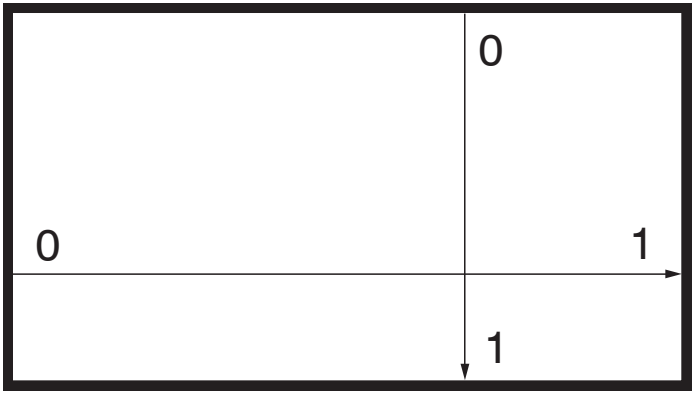


Figure 17. The infrared overlay emits position messages with x and y values ranging from 0 to 1.

This is not a problem for our calculations as long as the screen is a square. Unfortunately it is not. Our screens usually have the aspect ration of 16:9.

One should think of a stick that has the height of the screen. The y distance would be 1. If you now keep the lower end where it is and turn the stick by 90 degrees the stick is of course still as long as before, but the x distance the screen return is just ~0.56, because the stick is not as long as the full screen.

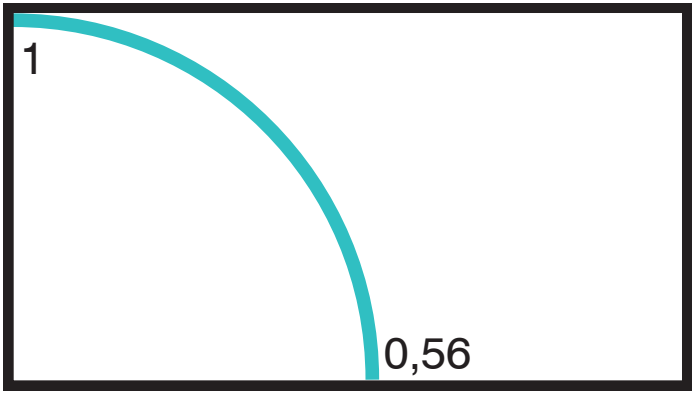


Figure 18. The same distance on horizontal and vertical axis is not represented by the same x and y values sent by the overlay.

When turning a tangible on the screen, angles and lengths vary in our data.

This is simply corrected by multiplying the x value of the position by the aspect ratio of the frame before executing calculations. In our case the y coordinates now vary between 0 and 1, the x coordinates between 0 and ~1.777.

The calculations are flawless with this correction.

2.3. How to prevent normal multi-touch gestures to be interpreted as tangibles

A challenge in the simultaneous use of normal touch gestures and tangibles that are recognized as fingers is the distinction between them. How is it possible to prevent normal touch gestures from being interpreted as tangibles?

Adjust the recognition tolerance

A human can produce up to 120 triangles with his 10 fingers. The possibility that some of them resemble triangles of the tangibles is quite high.

To lower this probability, the tolerance in side length difference accepted by the algorithm can be lowered.

The number of user-created triangles falsely recognized as tangibles decreases, but so does the number of correctly identified tangibles. This is caused by a known imperfection in the detection frame:

The overlay frame tends to shift already registered point positions as new points are detected.

To describe this effect further: A false finger is placed on the screen and from then on not touched anymore. If a another finger is now placed in proximity to the first and moved, it is possible to observe the detected position of the immobile first finger changing.

Allowing a higher tolerance also keeps the rate of recognized tangibles up. To a certain degree this is improving the recognition, but after a while this works in favour of the real fingers being able to “clone” patterns.

This peculiarity of the system interferes with using small tolerances to improve false detection rates.

Require more recognized triangles per tangible

Another approach in preventing real fingers from being detected as tangibles is to require a certain amount of triangles from the tangible to be detected.

This of course works only if the tangible consists of enough points to create that many triangles. A 3 point tangible can only create 1 triangle. A 5 point tangible already creates 10 triangles.

Require the recognition of the same tangible over a specific period of time

As the users keep their fingers moving on the display surface, opposed to the tangibles that are most of the times static, the probability that they manage to rebuild a tangible pattern over a period of several frames is quite low.

To take advantage of this, the implementation of a counter is useful. Only if the tangible has been recognized in a position for a couple of frames it is likely to be there.

Check for plausibility in movement

If the tangible is well recognized in one corner of the screen and then suddenly appears on the other side in the next frame, it is not likely to be correct.

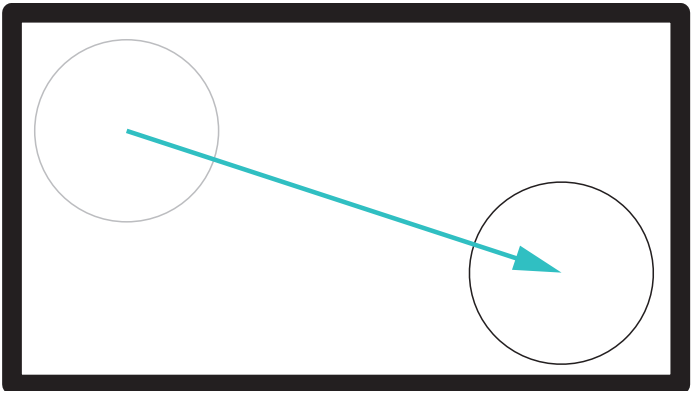


Figure 19. Large jumps in the tangible position are unlikely to happen and can be filtered by a low pass.

A simple low pass filter can prevent “large jumps” in the tangible position.

The tangible acceleration is also limited by this measure.

2.4. How to prevent tangibles to be interpreted as multi-touch gestures

The tangibles produce the same signal as real fingers. Moving a tangible over a touch button would trigger it.

Filtering the false fingers

If the proxy software does not forward the signals created by a tangible, they are not able to trigger the button.

All finger signals have to be checked for the possibility to be part of the tangible footprint and when they are likely to be a part of it not be forwarded.

This can be done by saving the IDs of the “live” cursors that were recognized to belong to the tangible. The list with those IDs is subtracted from the list of the current live cursors and the false fingers belonging to the tangible are not received by the user software anymore.

The problem with the user’s grip

Not only the “false” fingers of the tangible can trigger touch sensitive fields like buttons on the interactive surface. Also the human fingers holding the tangible tend to trigger a touch signal on the screen.

A simple solution is to create a circle around the center of the detected tangible. All cursors in this defined circle are removed from the buffer and therefore not visible by the receiving software.

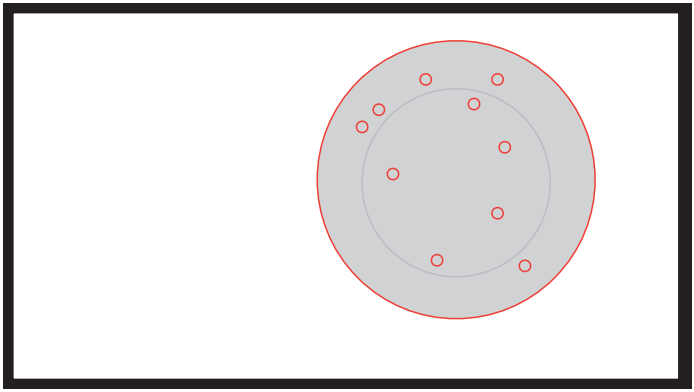


Figure 20. A circle to filter all cursors around and belonging to the tangible. This prevents grabbing fingers from triggering buttons.

With the use of this filtering option there is no need to compare the tangible IDs with the live cursor IDs. Even though it tends to be slower.

Place buttons relative to the tangibles

Another solution would be arranging buttons in relative position to the tangible. This way it is not possible to move over a button because the buttons are moving with the tangible.

The interface design needs to be adopted to this. This solution seems to be quite an inelegant correction.

2.5. How many different tangibles can be detected when the scanning resolution and quality is limited

A frame with a resolution of 6 points could detect two tangibles with 3 points, as long as no real finger is “involved” in this process, which is difficult to prevent.

As soon as there are more fingers (real and false) on the screen than the resolution provides, the whole system is likely to fail.

Due to this cause it is advised to use overlays with 32 or more recognizable points.

3. Infrared overlay

The infrared overlay is quite similar to a picture frame. It is placed on top of a screen or even just a flat surface.

3.1. A simple grid

Infrared diodes and sensors are integrated in this frame and build a grid.

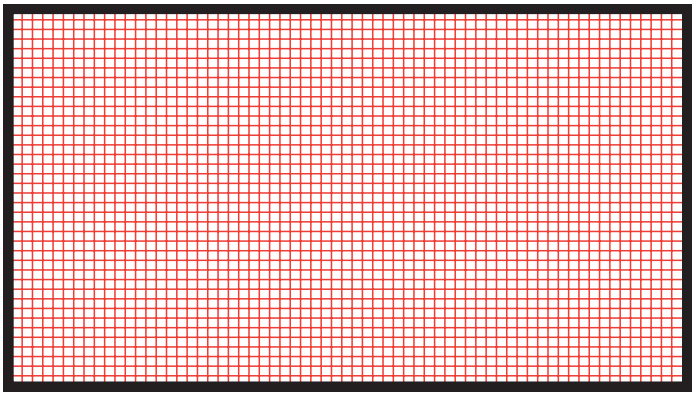


Figure 21. Schematic of the infrared grid

When this grid is disturbed by a finger, or a something else, the frame hardware passes this signal over USB on to a parsing software which then sends TUIO messages to the interactive software.

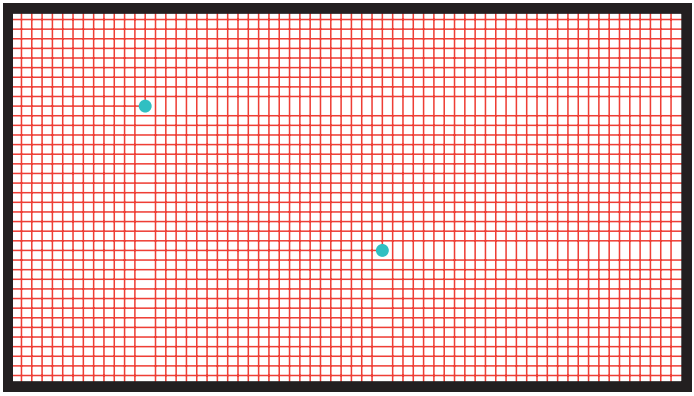


Figure 22. Grid disturbed by objects e.g. finger. This would lead to 4 recognized points. Two of them "ghost points".

This is of course a simple way of getting positions. And would also lead to 4 recognized points. The above described way of recognition should just give you an idea of how those frames work.

A more sophisticated solution is the the following:

On one side of the overlay are infrared diodes aligned. On the opposing side are infrared sensors.

Figure 23. Overlay with infrared LEDs (red) and infrared sensors (blue)

One diode after another is switched on and emits light into the direction of the sensors.

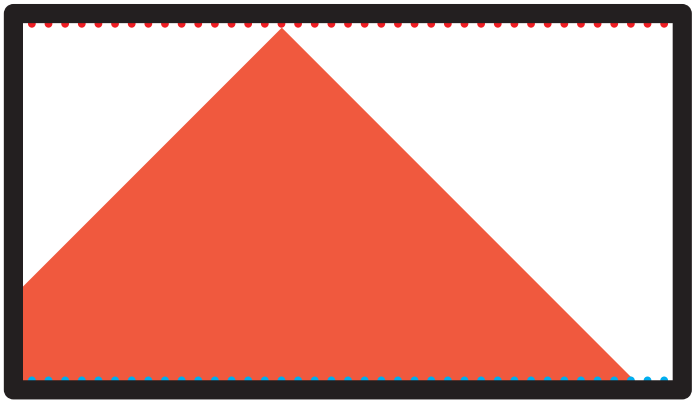


Figure 24. One diode emitting infrared light.

An object in this light casts a shadow on the sensors. The area between the non-lit sensors and the emitting diode is a first hint on the position and size of an object on the screen.

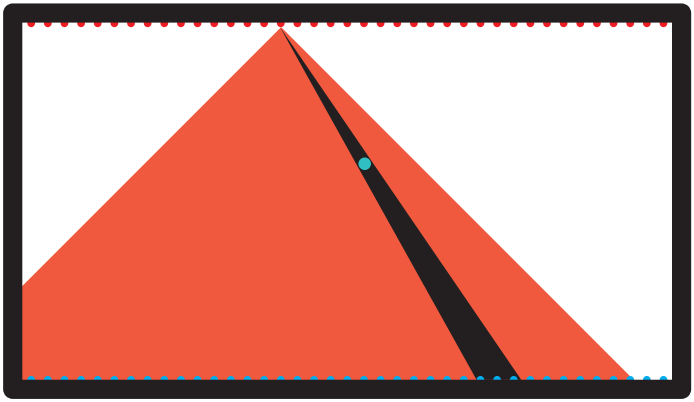


Figure 25. An object creating a readable shadow on the sensors. The black area is a hint on the size and position of the object.

By overlaying several shadow areas, that have been optained seperately, the position and shape of an object is narrowed down.

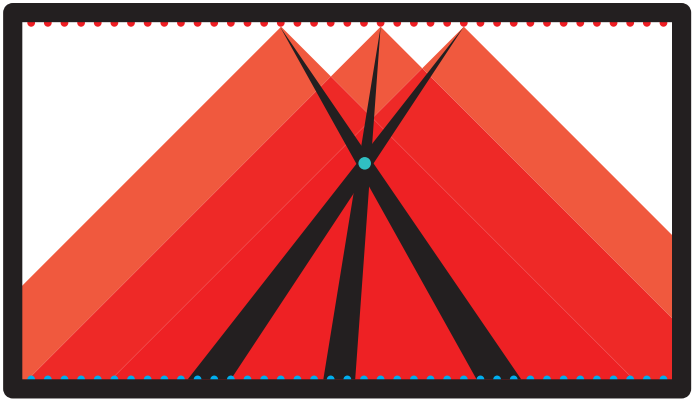


Figure 26. Overlay of shadow areas

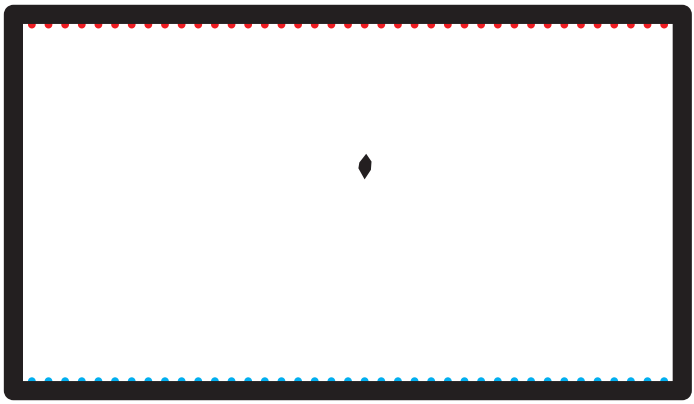


Figure 27. The intersection of all shadow areas narrows down the shape and position

The recognized shape improves with the amount of used LEDs and sensors.

By adding the same system on the two other sides of the overlay, the system can be improved even more.

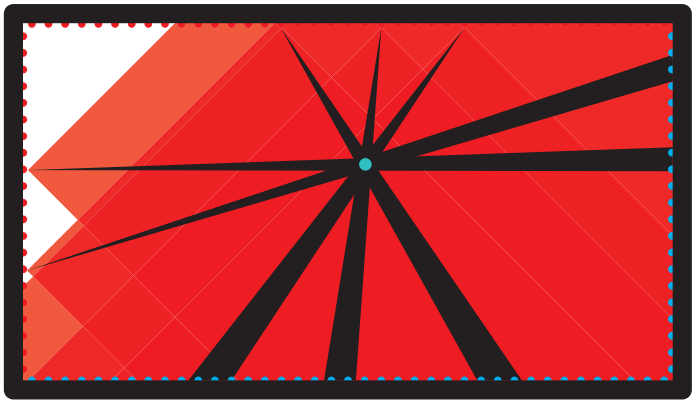


Figure 28. Use of a vertical and horizontal recognition for the recognition

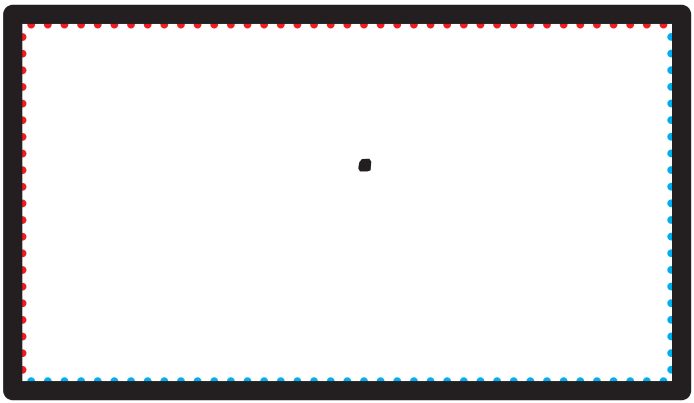


Figure 29. Result when horizontal and vertical recognition are combined

For further details visit the [PQ Labs] website or have a look at [US Patent Application 20120098753] from Fei Lu to get a closer look on how this frame works in detail.

Even more complex calculations with interferences and superpositions come to ones mind and depend on the hardware manufacturer.

The used hardware for the tests has been a PQ Labs G3 55” overlay with 32 points.

3.2. Screen protection

To protect the underlying screen it is advised to place a glass surface between the screen and the overlay.

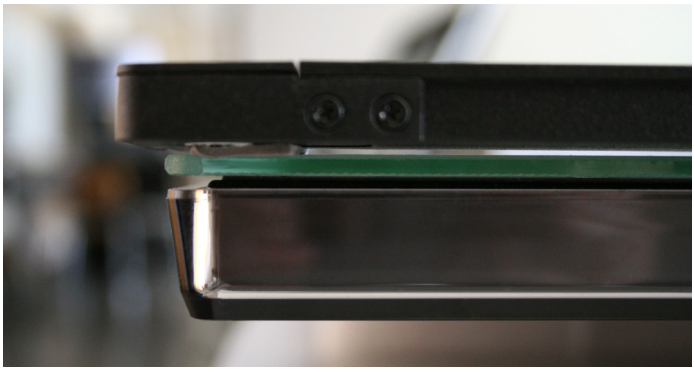


Figure 30. Screen, protective glass and multi-touch frame

4. The tangibles

The basic structure of the used tangibles is quite simple:

- Small objects that can be recognised as fingers by the multi-touch system.
- An object on top of those false fingers to keep them in the same position towards each other and to give the user the ability to control the tangible in its position.

4.1. The false fingers

Shape

The false fingers should be round in shape to have always the same appearance towards the frame, regardlessly on their rotation. This way their center is always the same.

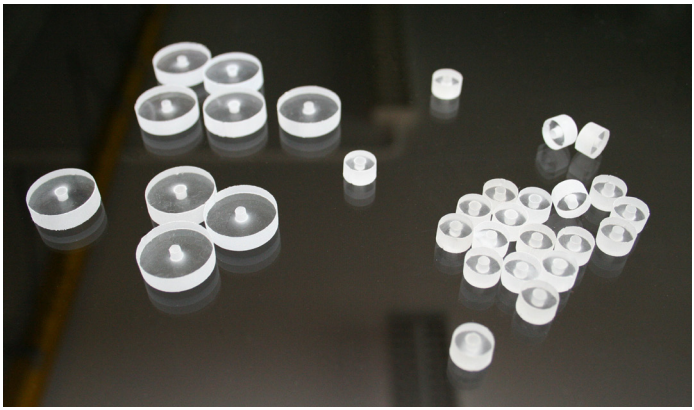


Figure 31. False fingers with drilled holes for fixation

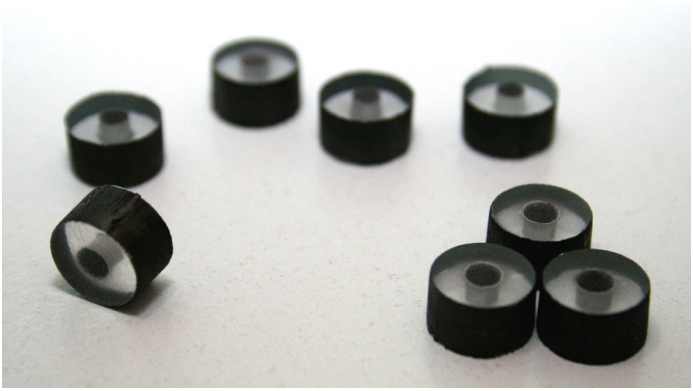


Figure 32. Rounds with black coat to prevent reflections and refractions

Detectable size of false fingers

Tests have been started with 21 mm in diameter and been narrowed down to 10 mm. A diameter smaller than this is most of the time not recognized as objects. 5 mm rounds are never recognized.

This behavior depends on the model and manufacturer of the overlay and the configuration files of the hardware vendors parser.

Minimal distance of the floating surface over glass

“Non-finger” parts of the tangible need to have at least 5 mm distance to the surface for not being detected by the frame.

This is specific to the hardware used as it depends on height of the LEDs and sensors above the touch surface.

4.2. The tangible shape

The tangible shape needs to fulfil several purposes:

- The tangible should be tangible by human hands, preferably by a single hand.

- It should bear the possibility to spread the false fingers underneath it in that way that a large distance between them is possible. This improves the insensitivity towards errors in position recognition by the overlay

A prototype made from a compact disc showed that its size is a good one to use:

- It gives the possibility to have different false finger positions underneath it and it is possible to easily place 5 fingers on top of them.

- Its round shape is quite universal.

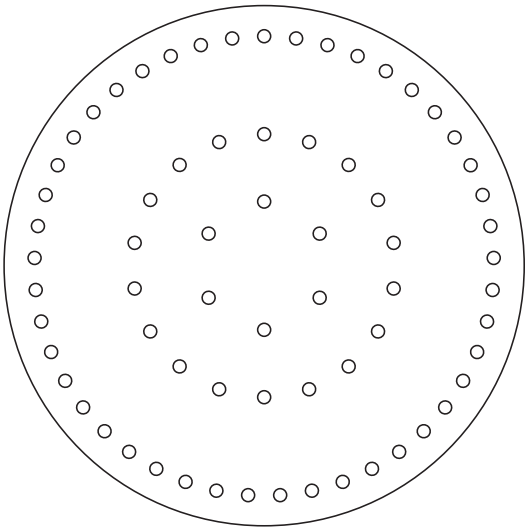


Figure 33. A round tangible with the opportunity to fix false fingers in the center

This shape can be altered with a hole in the middle.

This provides the possibility to have an image in the middle of the tangible and also to use this area as a touch button.

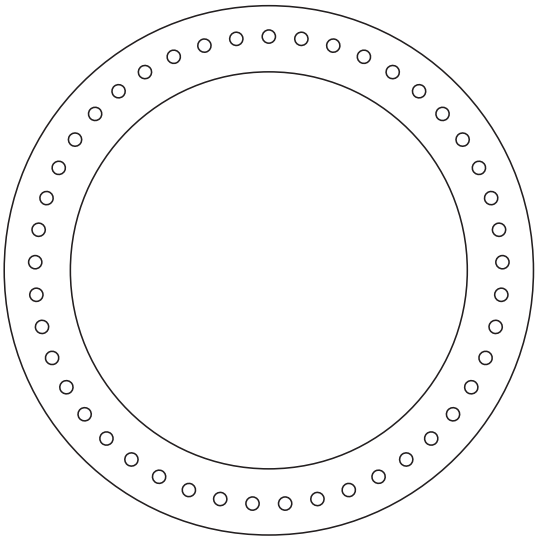


Figure 34. A round tangible with a big hole in the middle

Those two options can also be used with a non-round shape, like an rectangle.

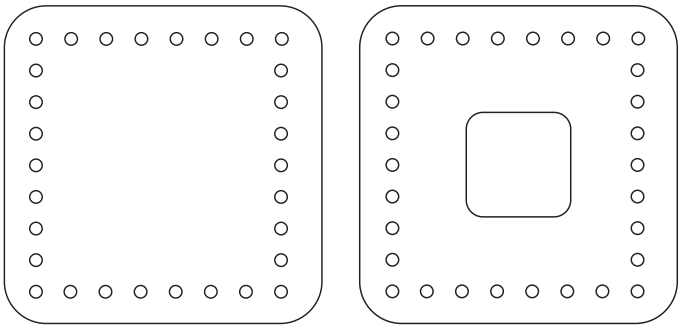


Figure 35. Two almost rectangular tangibles, the right one with hole.

An option is also having them in bigger sizes e.g. to move a separate screen on the display.

From there it is one step to just have a long ruler like bar to use it as base for pictures displayed on top or to attach some controls around it.

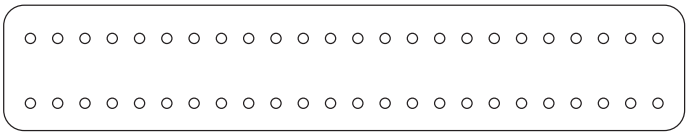


Figure 36. Long tangibles can be used as basis to layout buttons

Two elbow shaped tangibles can be used to span a picture between them.

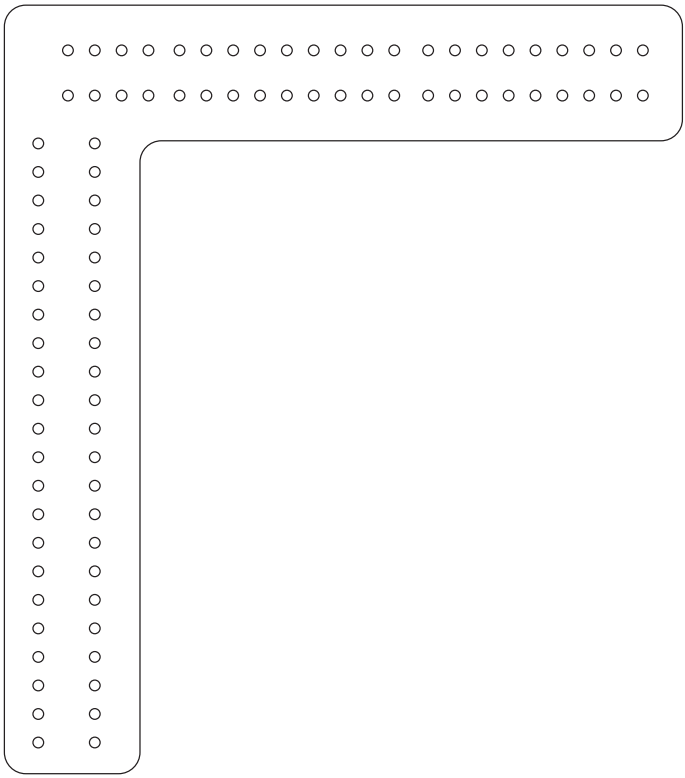


Figure 37. An elbow tangible. Two of them could be used to span a virtual screen.

4.3. Pictures of tangible prototypes

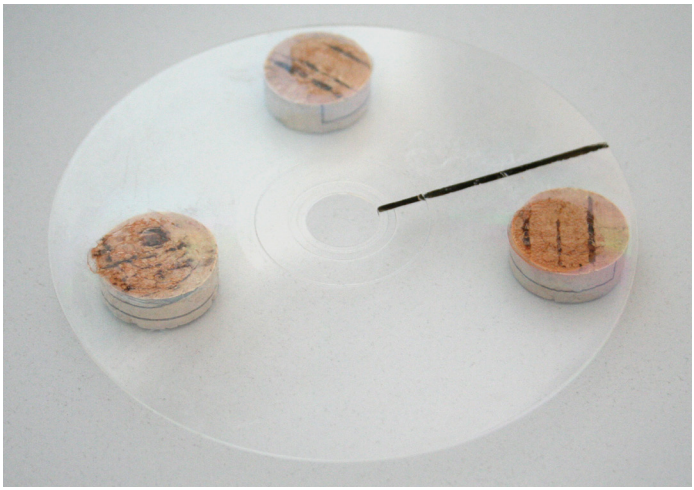


Figure 38. A tangible prototype consisting of cut cork and a CD dummy

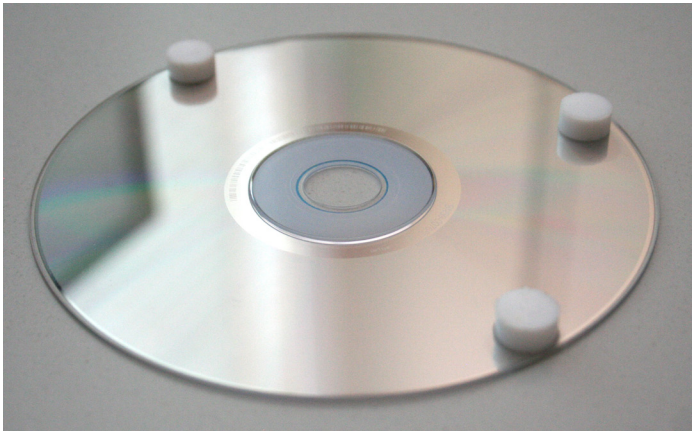


Figure 39. A tangible build of a CD and Teflon rounds

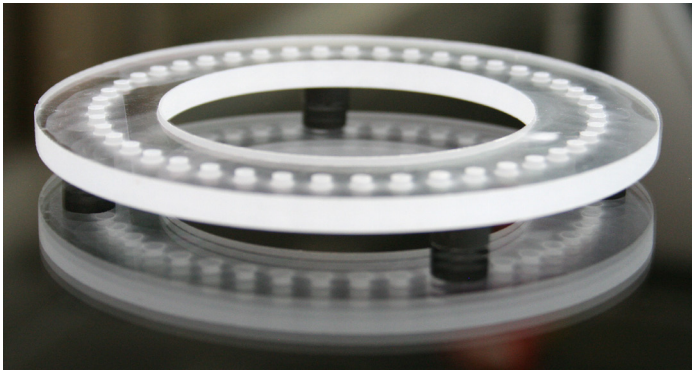


Figure 40. A tangible cut by a water jet. The hole in the middle allows the user to trigger buttons inside the tangible.

5. TUIO proxy

The proxy software is the program that receives the signals emitted by the infrared overlay's parsing software. In those messages it looks for tangible patterns and, if recognised, sends messages with position and rotation of the tangibles to the client software.

5.1. What it does

Normal message flow

When a user touches the screen the infrared overlay detects the disruptions in its infrared grid.

The data generated by the frame is sent via USB to the parsing software of the overlay vendor. This software creates special network messages with the position of the touches which are then received by the software with the user interface.

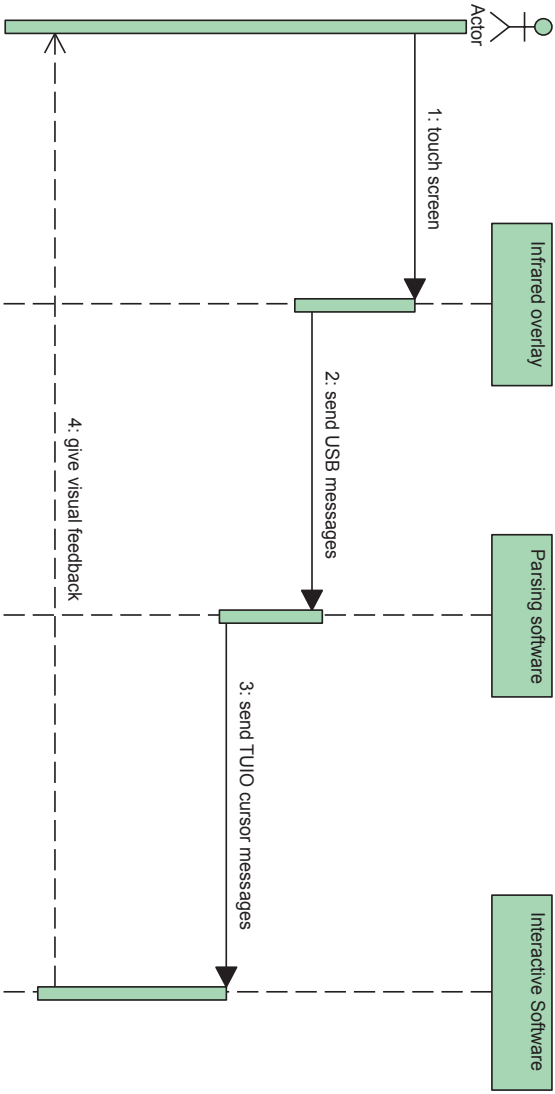


Figure 41. Usual data flow for multi-touch interactions

Message flow with proxy

A new program is inserted in the data stream. It takes a look into all the cursor messages sent from the parsing software.

If it detects a known pattern of cursors it inserts additional message in the cursor stream. Those “object messages” contain the ID, position and rotation of the tangible.

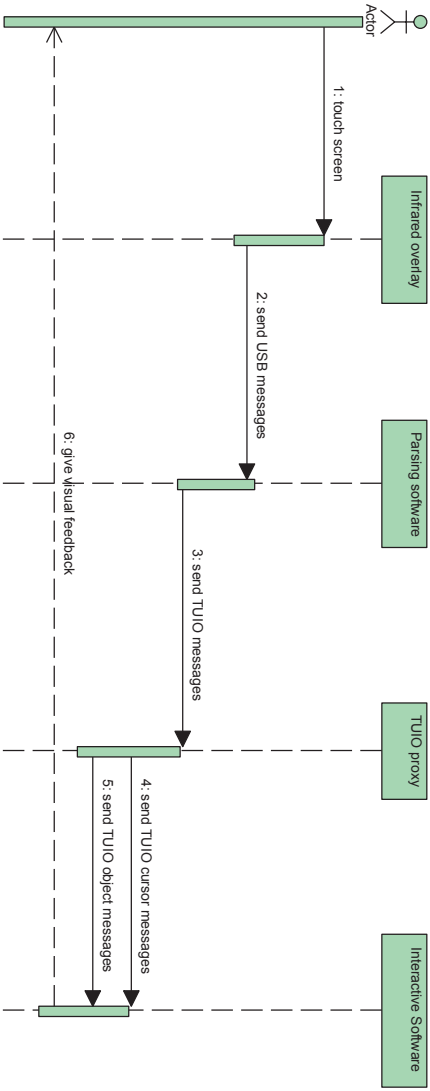


Figure 42. The multi-touch data flow through the proxy

5.2. How it works in detail

The proxy software sends and receives TUIO messages. TUIO is based on Open Sound Control, a protocol created to function as interface between multimedia devices and software. [Wright, M., Freed, A., Momeni A. (2003)].

TUIO protocol

Basically the TUIO protocol transports information about tangible positions, rotations and of course simple touches towards applications that then react to those inputs. One can consider it as mouse or finger positions transferred over a network protocol.

To group and specify what is going on the touch-surface several message profiles are defined. The messages mainly contain a position and acceleration.

For our purpose two message profiles are used:

- 2DCur - A two dimensional cursor (e.g. a finger touch)
- 2DObj - Same as a 2DCur, but with more information like rotation and ID

A 2DCur message looks like this:

```
/tuio/2Dcur set s x y X Y m
```

- s = session ID
- x,y = position
- X,Y = velocity vector
- m = motion acceleration

A 2DObj message looks like this:

```
/tuio/2Dobj set s i x y a X Y A m r
```

- s = session ID
- i = class ID
- x,y = position
- a = angle
- X,Y = velocity vector
- A = rotation velocity vector
- m = motion acceleration
- r = rotation acceleration

Both message types are by standard bundled with a previous alive message giving the IDs of the cursors that are still active. The message looks like this:

```
/tuio/2Dcur alive s_id0 ... s_idN
```

At the end a message with an incrementing sequence number is attached:

```
/tuio/2Dcur fseq f_id
```

In total a message bundle would look like this:

```
/tuio/2Dcur alive s_id0 ... s_idN

/tuio/2Dcur set s_id x_pos y_pos x_vel y_vel m_accel

/tuio/2Dcur fseq f_id
```

For further information, please visit [tuio.org] where also the previous specification examples are taken from.

5.3. The recognition and calculation

The recognition and calculation basically follows the scheme described in the algorithm chapter.

The following diagrams show the significant code parts as UML diagrams.

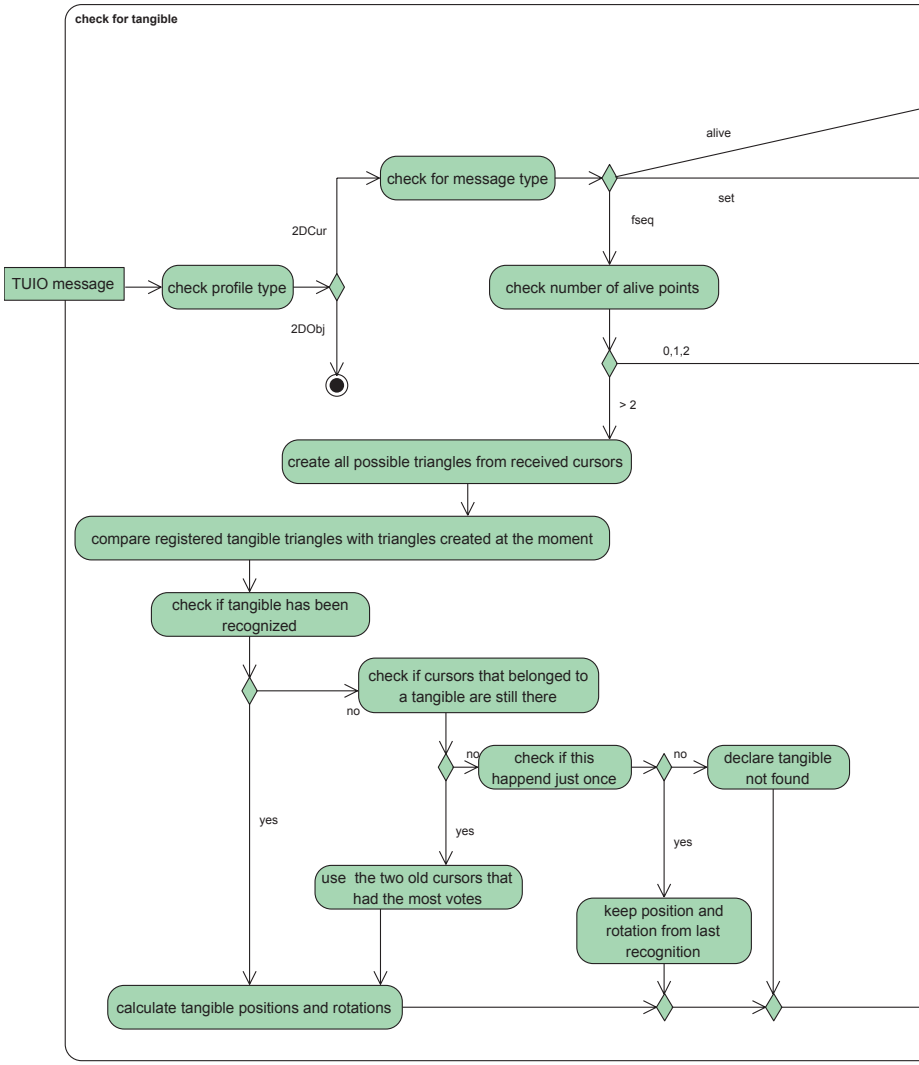
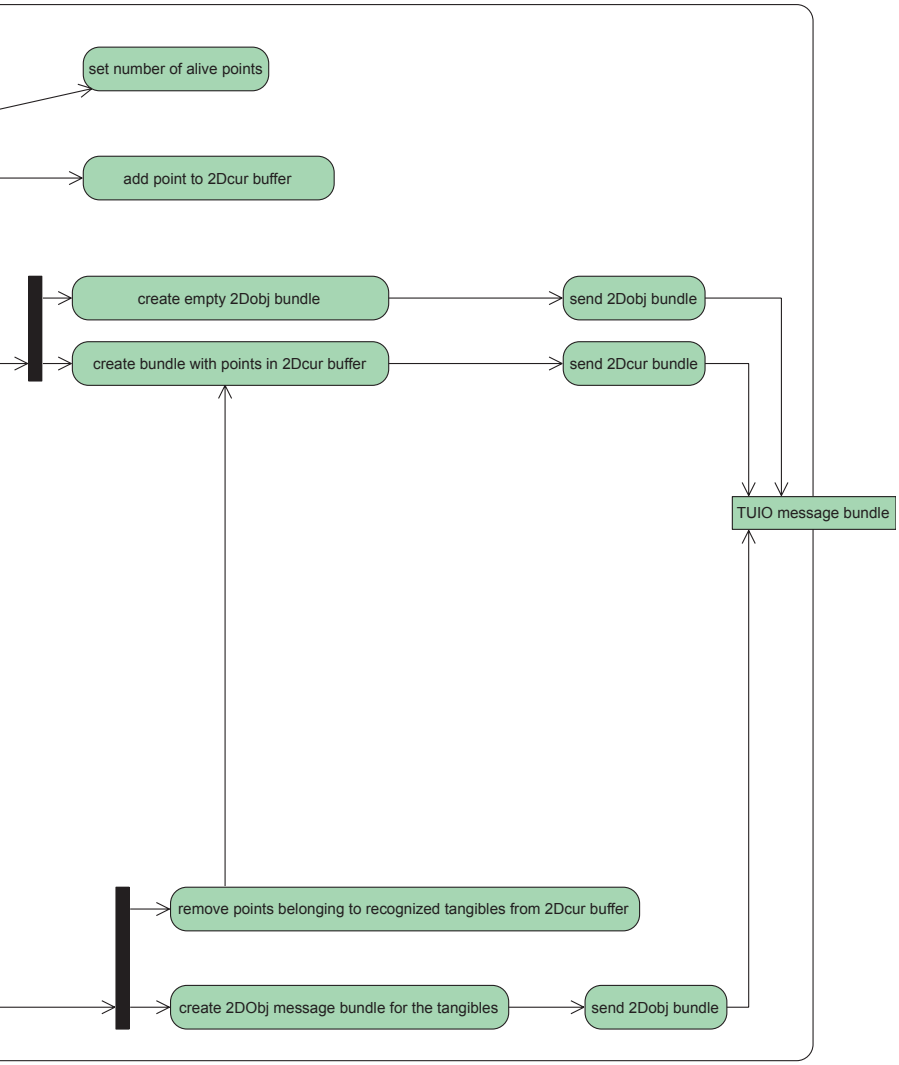


Figure 43. Schematic overview of the TUIO message handling


set number of alive points
add point to 2Dcur buffer
create empty 2Dobj bundle
send 2Dobj bundle
create bundle with points in 2Dcur buffer
send 2Dcur bundle
TUIO message bundle
remove points belonging to recognized tangibles from 2Dcur buffer
create 2DObj message bundle for the tangibles
send 2Dobj bundle

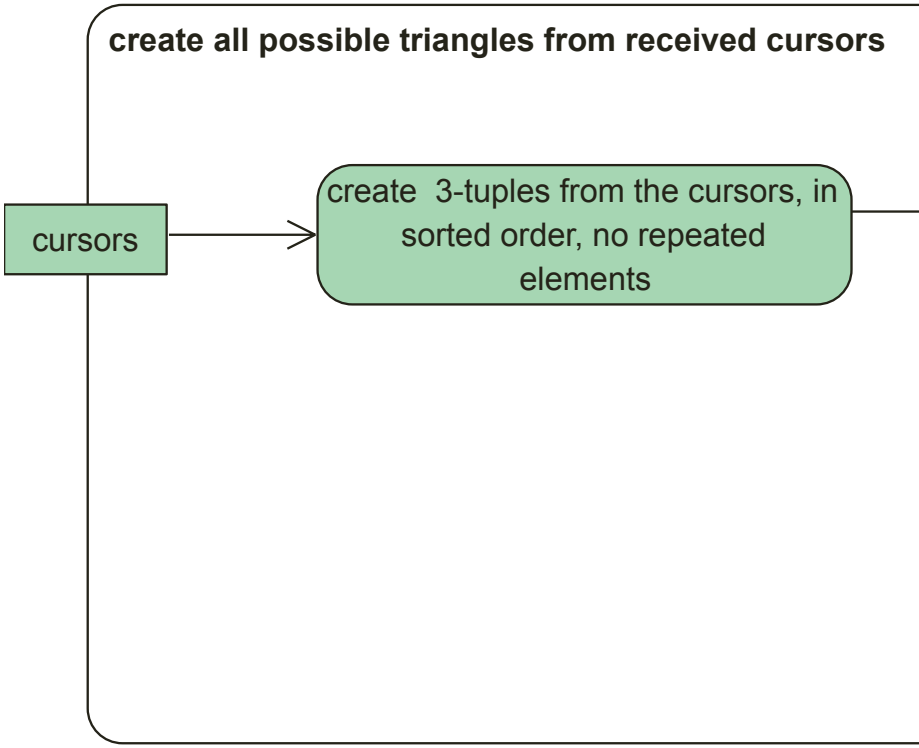
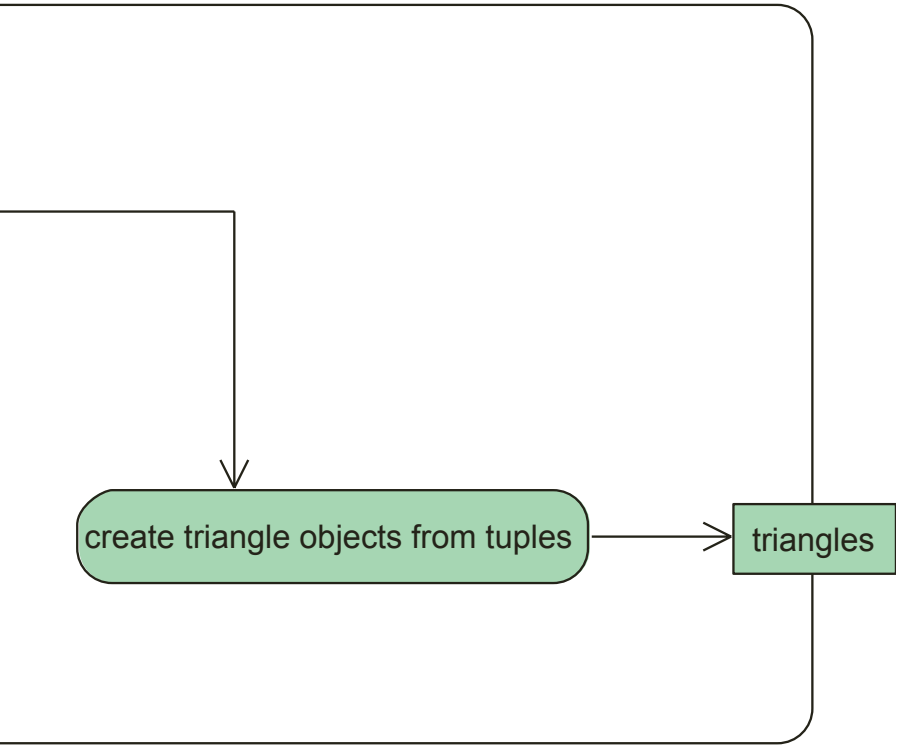


Figure 44. Creating all possible triangles from the received cursors


create triangle objects from tuples
triangles

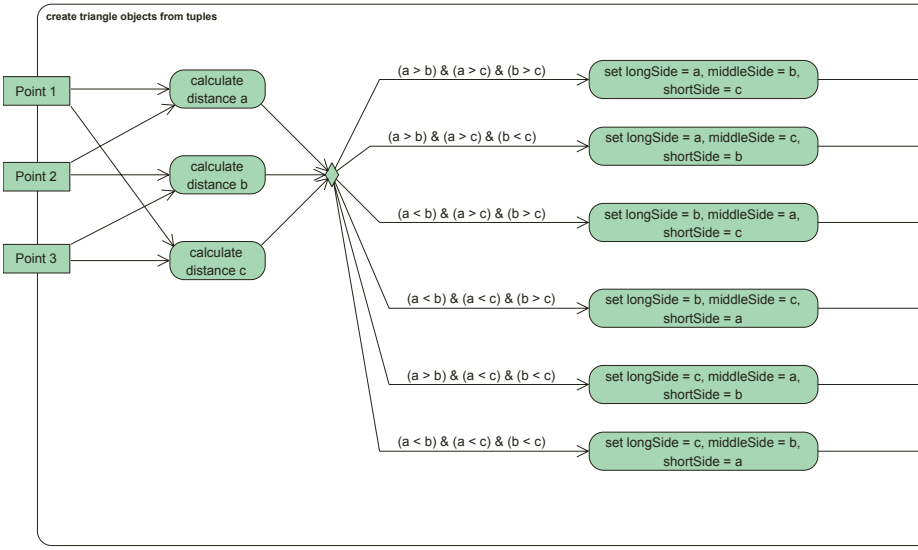
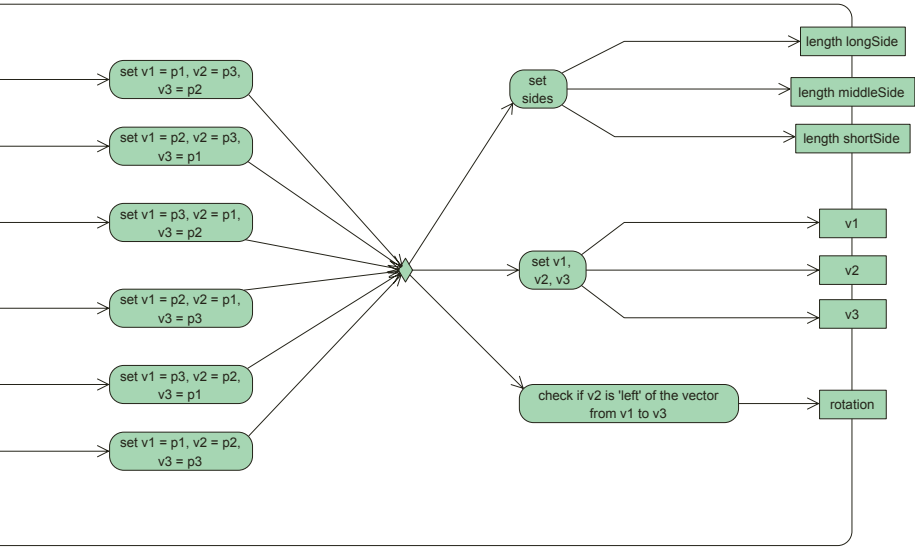


Figure 45. v1-v2 the short, v2-v3 the intermediate, v3-v1 the long side


set v1 = p1, v2 = p3, v3 = p2
set v1 = p2, v2 = p3, v3 = p1
set v1 = p3, v2 = p1, v3 = p2
set v1 = p2, v2 = p1, v3 = p3
set v1 = p3, v2 = p2, v3 = p1
set v1 = p1, v2 = p2, v3 = p3
set sides
set v1, v2, v3
check if v2 is 'left' of the vector from v1 to v3
length longSide
length middleSide
length shortSide
v1
v2
v3
rotation

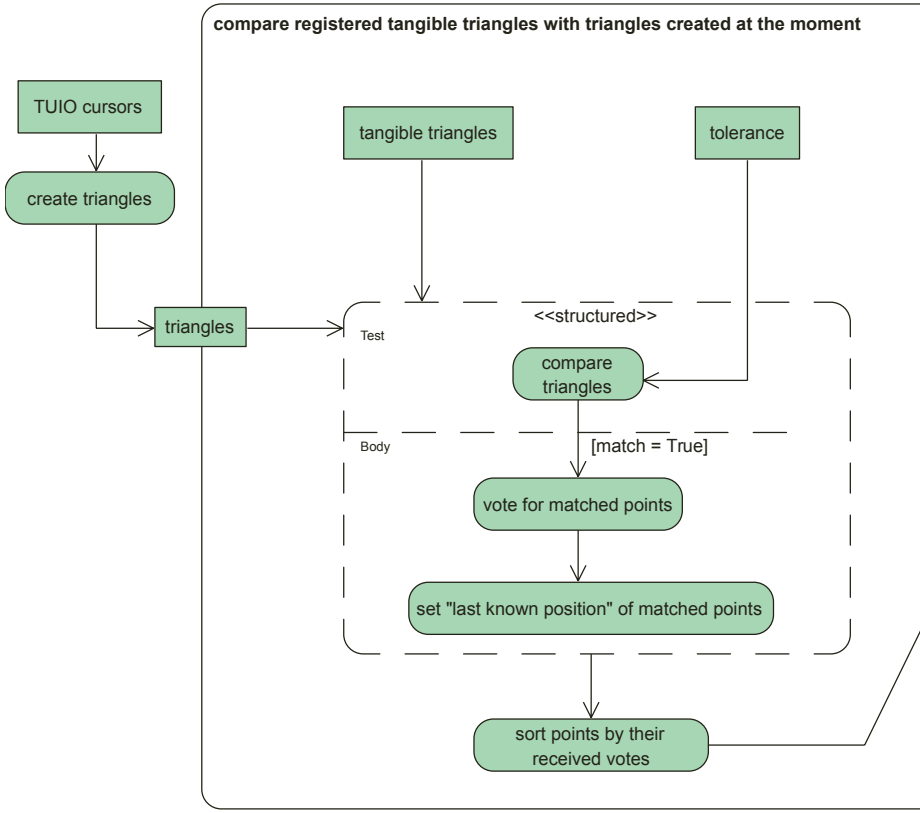
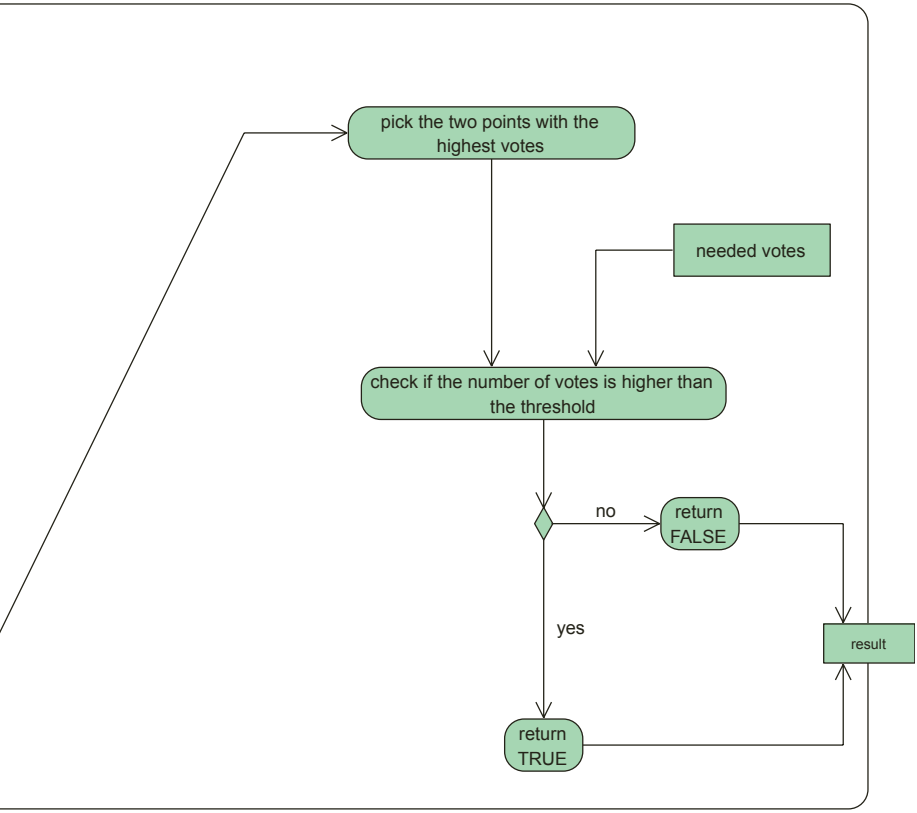


Figure 46. Compare tangible triangles and triangles of momentary


pick the two points with the highest votes
needed votes
check if the number of votes is higher than the threshold
no
return FALSE
yes
result
return TRUE

cursors

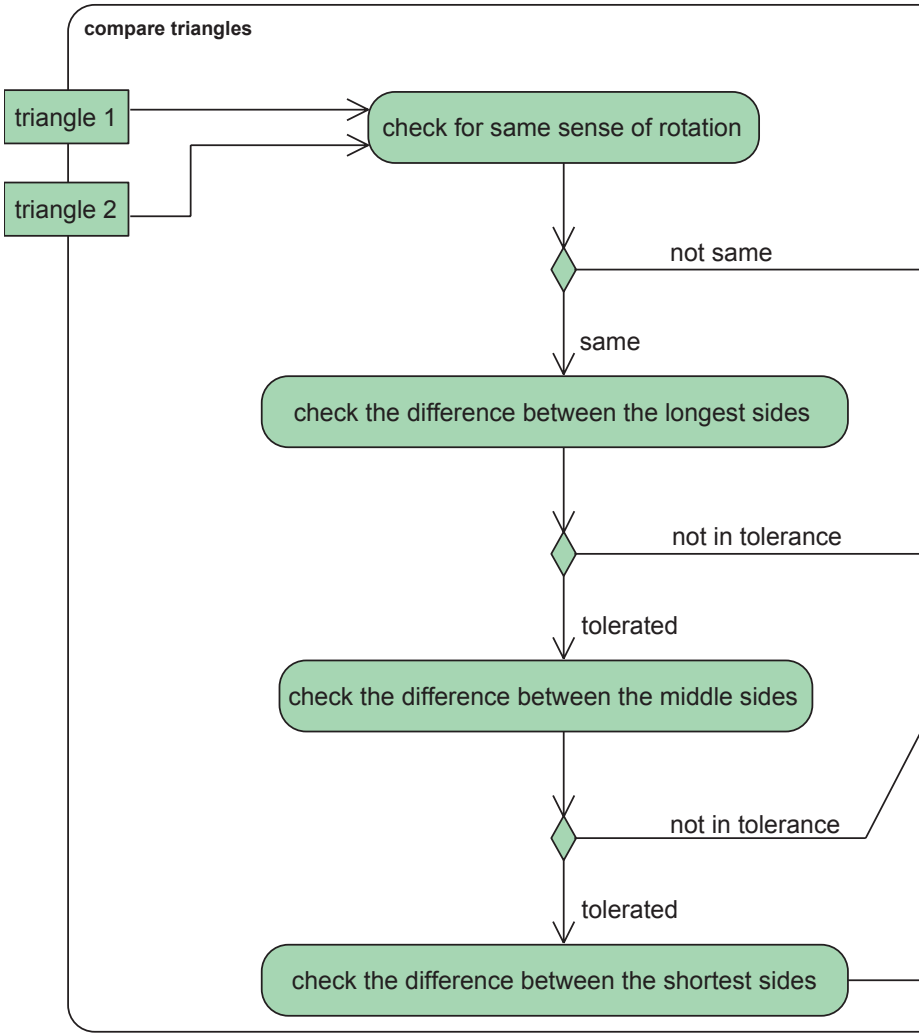
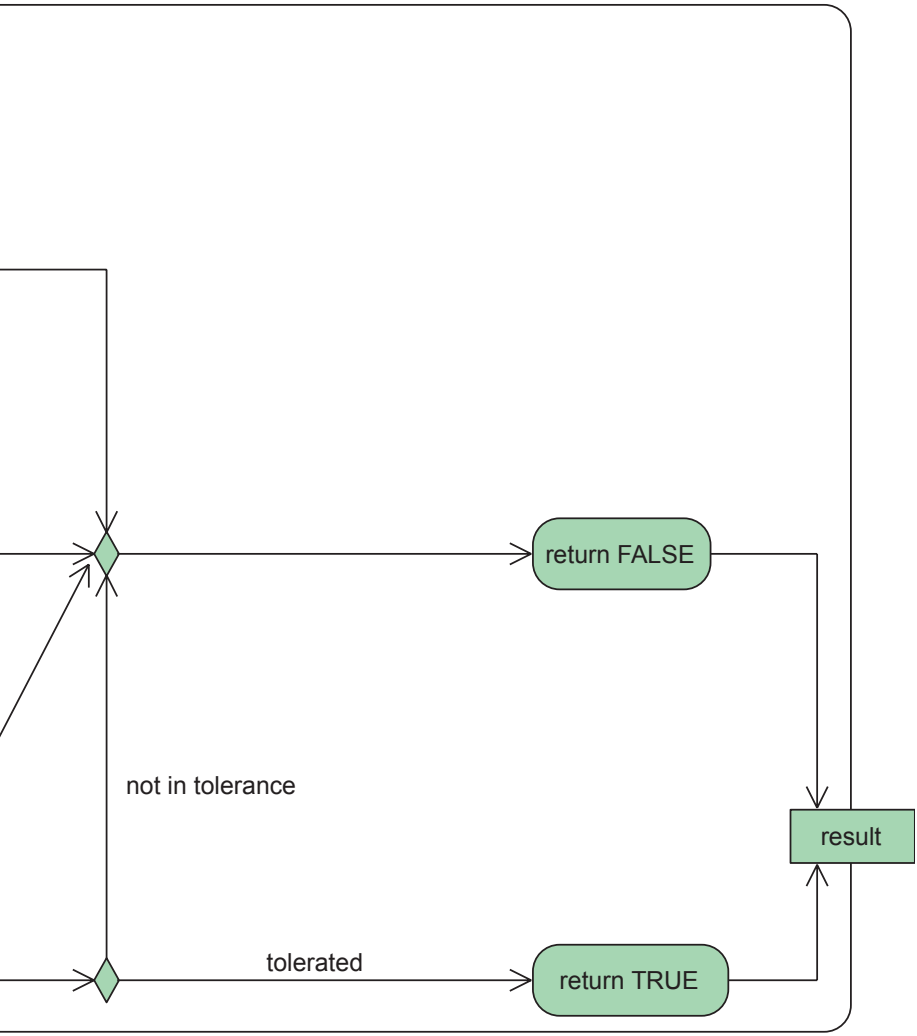


Figure 47. Checking two triangles for similarity


return FALSE
not in tolerance
result
tolerated
return TRUE

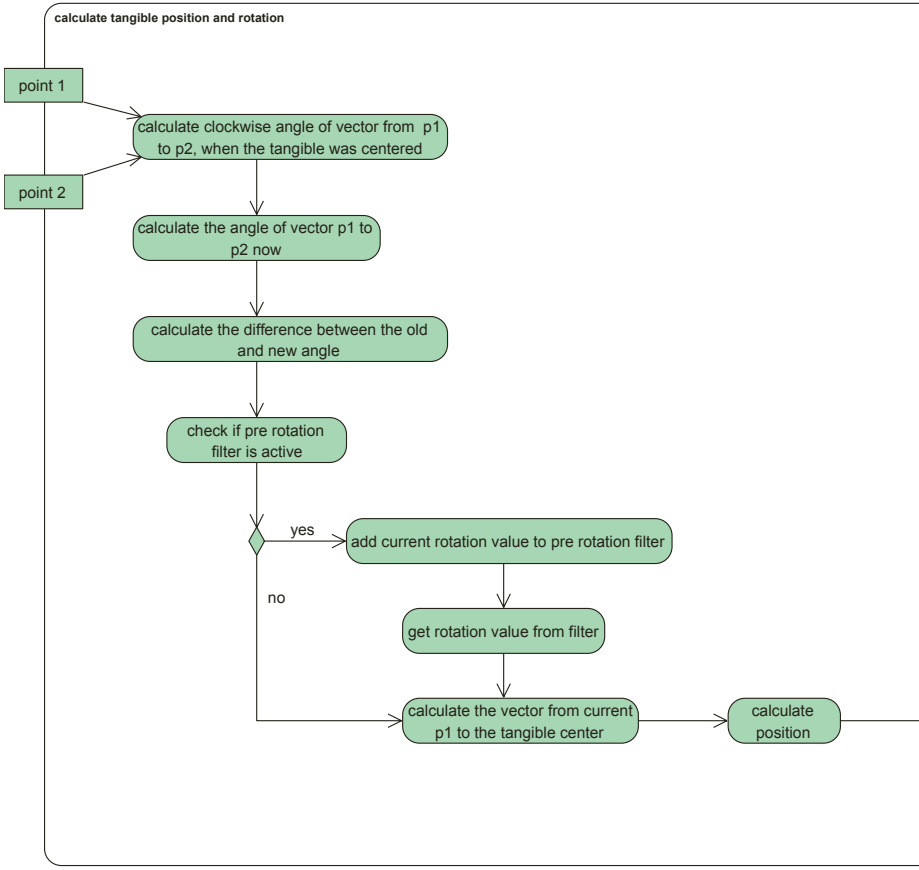
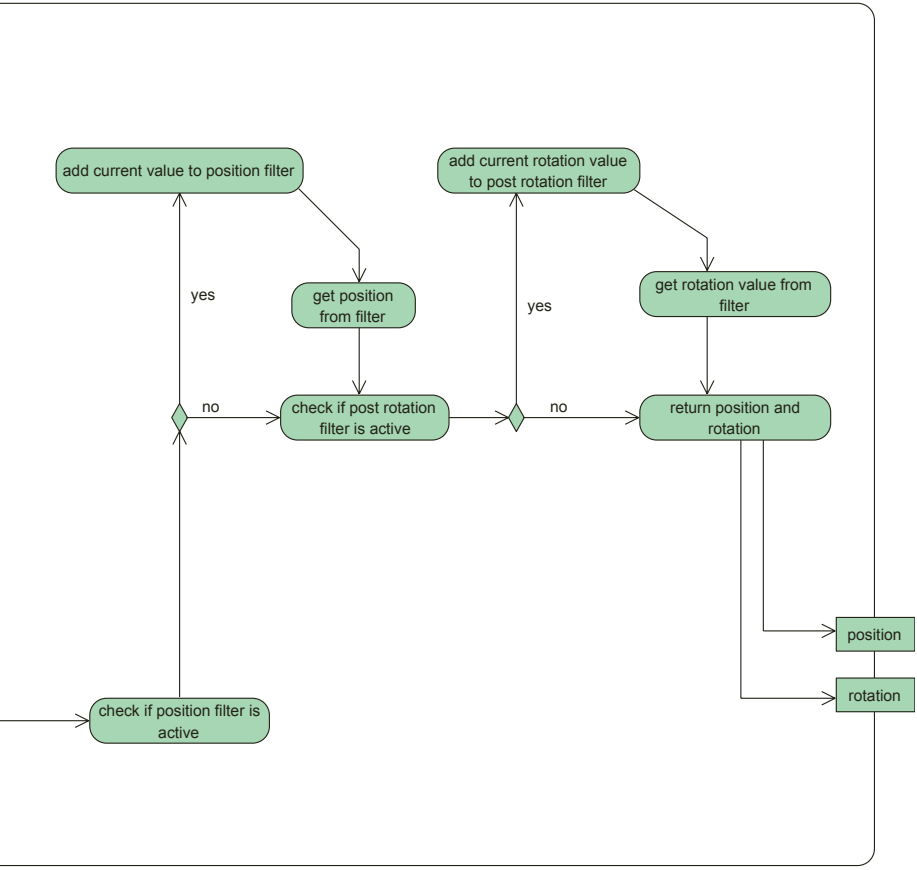


Figure 48. Calculate the tangible position and rotation after detection


add current value to position filter
yes
get position from filter
no
check if post rotation filter is active
add current rotation value to post rotation filter
yes
get rotation value from filter
no
return position and rotation
position
rotation
check if position filter is active

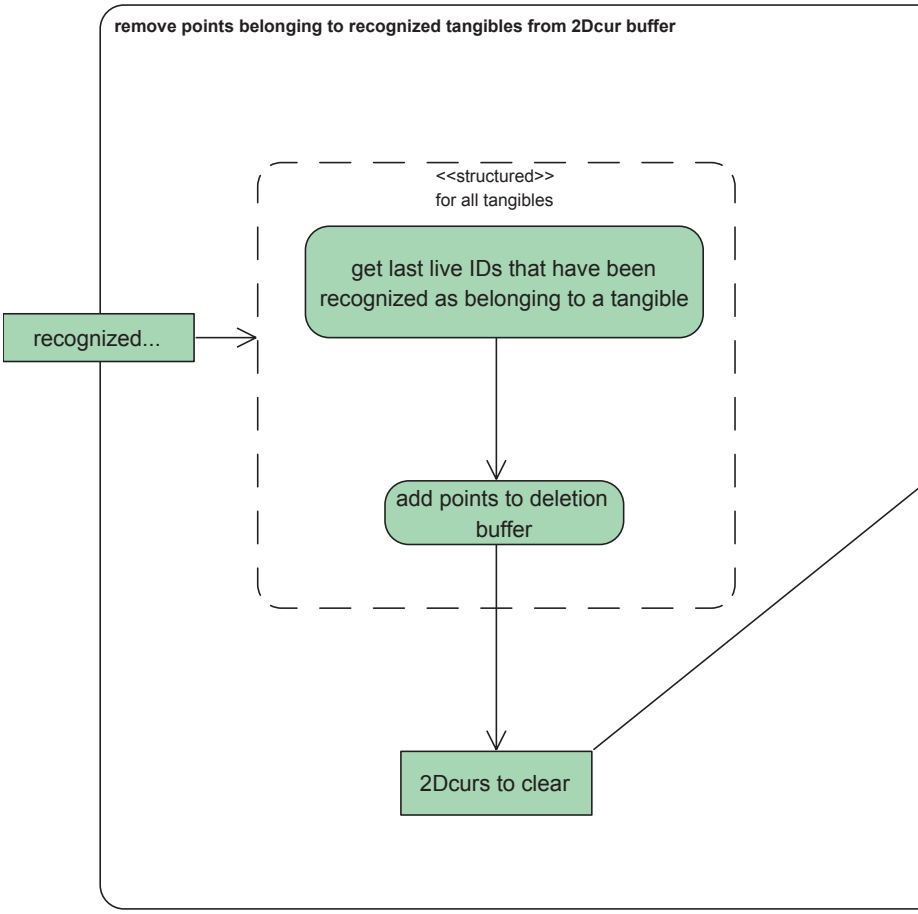
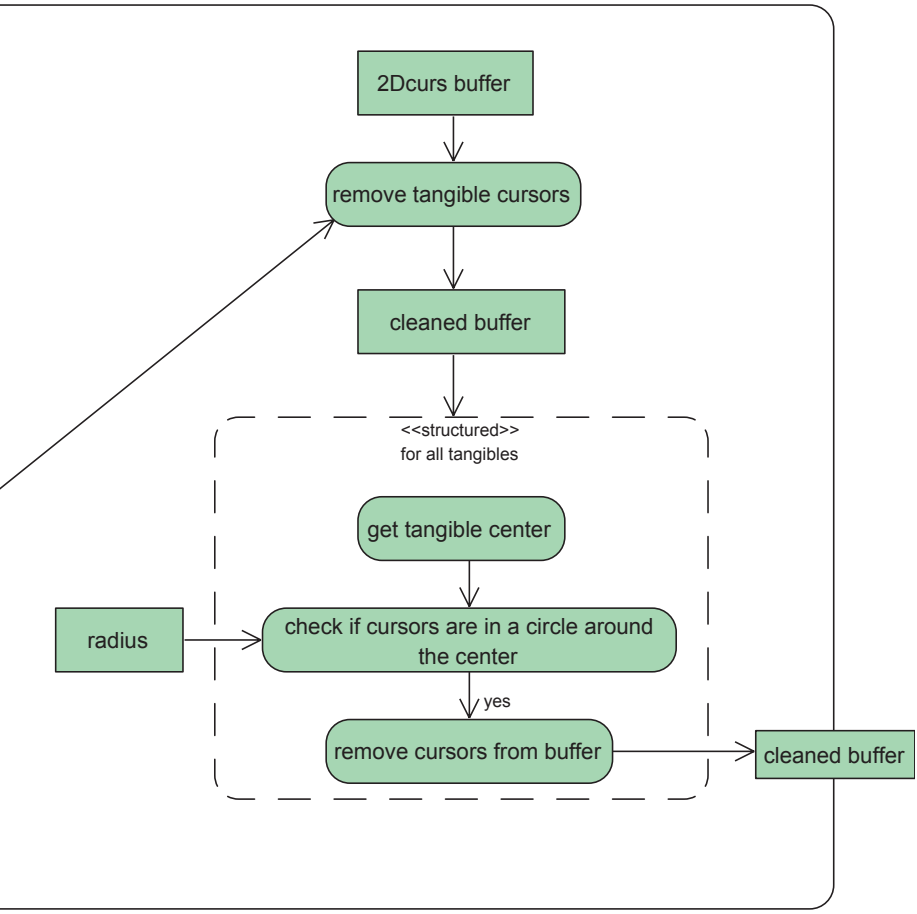


Figure 49. Remove points belonging to recognized tangible from cursor buffer


2Dcurs buffer
remove tangible cursors
cleaned buffer
<<structured>>
for all tangibles
get tangible center
radius
check if cursors are in a circle around the center
yes
remove cursors from buffer
cleaned buffer

6. Proxy control software

6.1. What it does

The proxy software is just a command line program without any graphical interface.

But it has a lot of settings that can be changed to adapt the algorithm and filters. To change those settings in a config file would be pretty cumbersome . Therefor a GUI has been created.

The GUI is not only a interface in which settings can be changed, it also displays TUIO cursors and TUIO objects. With this feature it is possible to directly see the effects of changes in the settings.

The GUI connects to the proxy via OSC. All settings are transmitted with this protocol.

All settings can be adjusted with sliders or buttons.

The program has been protoyped in Processing, due to easy accessible OSC, TUIO and GUI libraries.

6.2. Usage

After the program has been started it presents just a welcome message advising the user to open the tabs on top of the window in their given order.

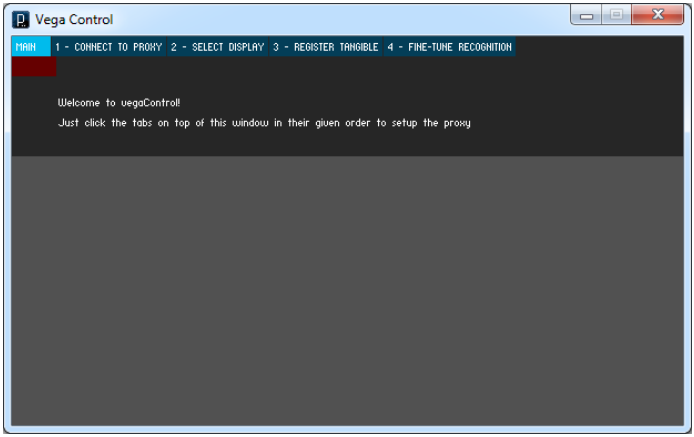


Figure 50. Welcome window of the control software

Connect to proxy

The first step is to connect the GUI with the proxy. This normally works automatically, but the proxy might be running on another machine and so the IP has to be altered.

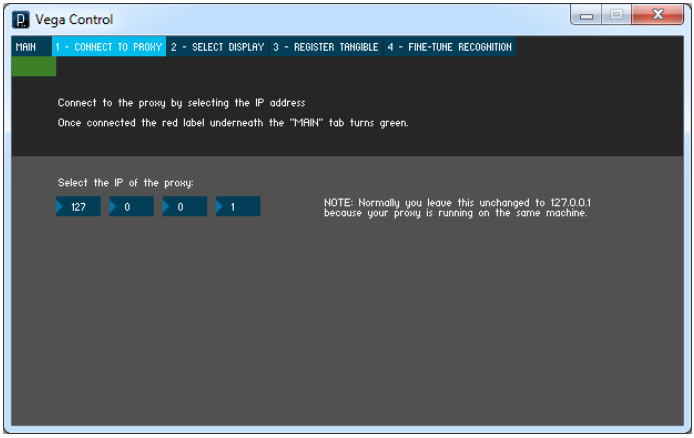


Figure 51. Connect to proxy - tab

Most of the time you would leave the IP unchanged.

You can see that the connection indicator above the main tab changed from dark red to green, this indicates that GUI and proxy are connected.

Select Display

In the next step the display for the further configuration is selected.

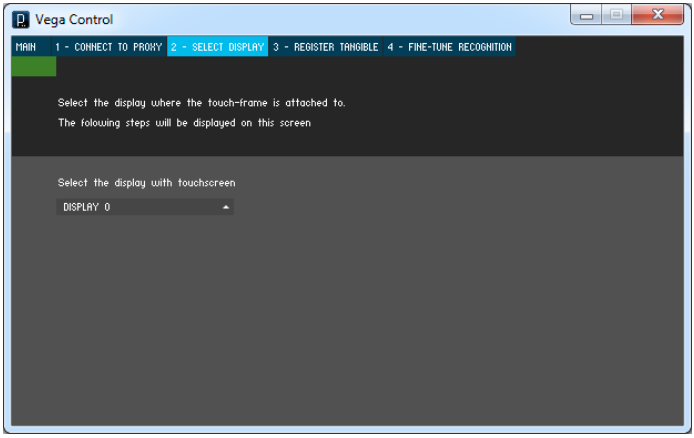


Figure 52. Selection of the display where the next steps are to be shown

This can be needed when you use a multi-monitor setup and your touch overlay is for example attached to a TV on a second display output. You want to display all the touches and TUIO cursors there, not on your main screen.

Register tangible

The next tab lets you register the tangible you want to use.

To see TUIO cursors and objects, activate the two buttons “show TUIO cursors” and “show TUIO objects”

The tangible has to be placed centered on the screen. A pattern helps you to see where the center is.

Now an ID is chosen by entering it in a textfield. A click on “Register tangible” saves all the features and the tangible is registered.

A tangible with a certain ID can also be deleted or you might choose to delete all of them.

All tangibles in the proxy's memory are lost the moment the proxy has been terminated, unless you click on the button "save to disk". In this case the tangibles are loaded again on the next start up of the proxy.

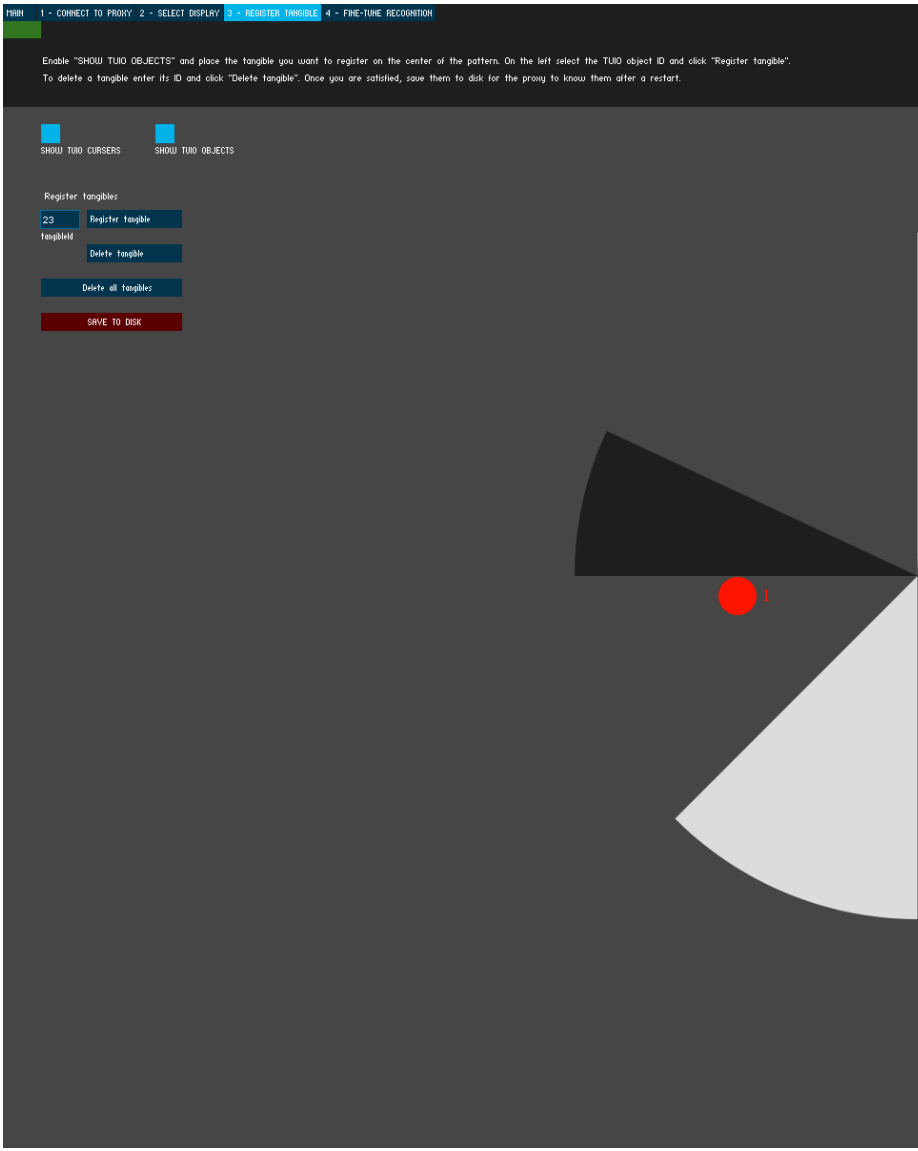
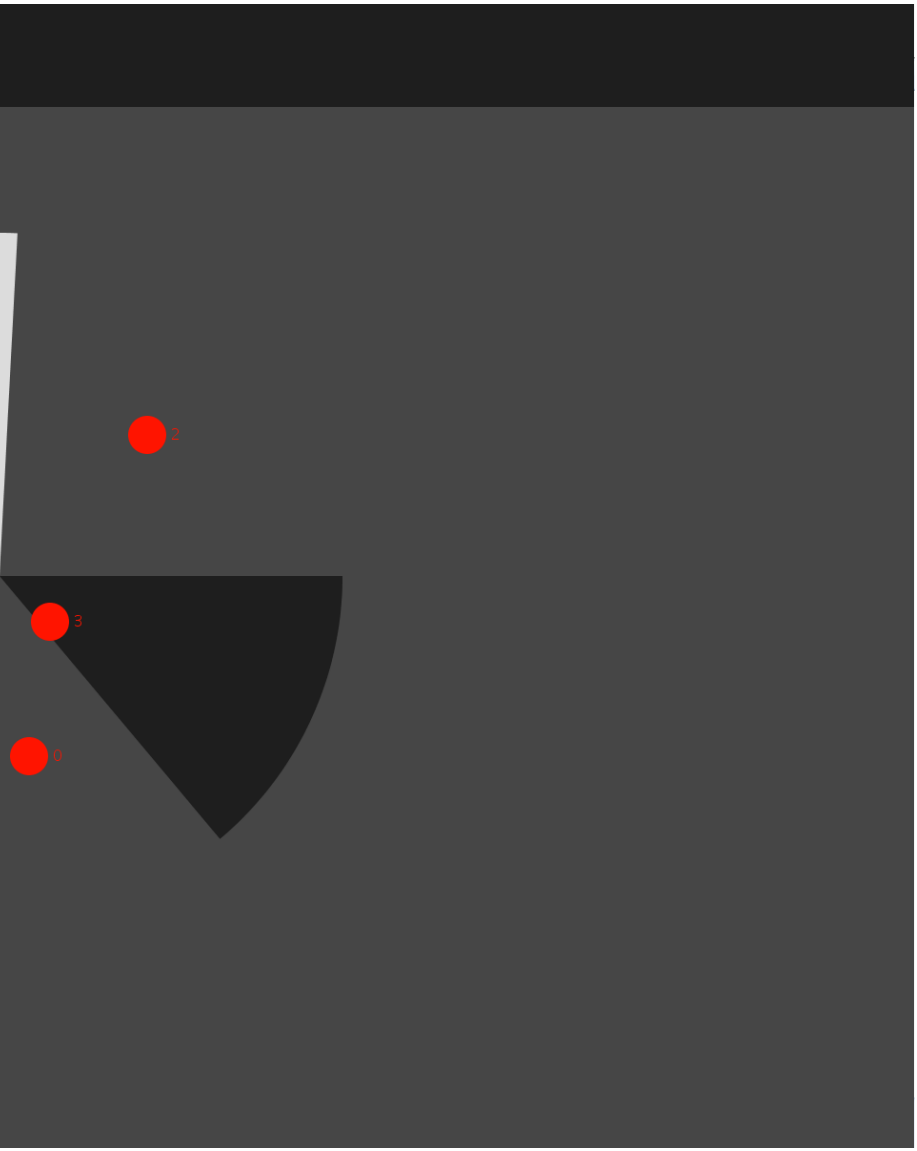


Figure 53. The tangible registration tab


2
3
0

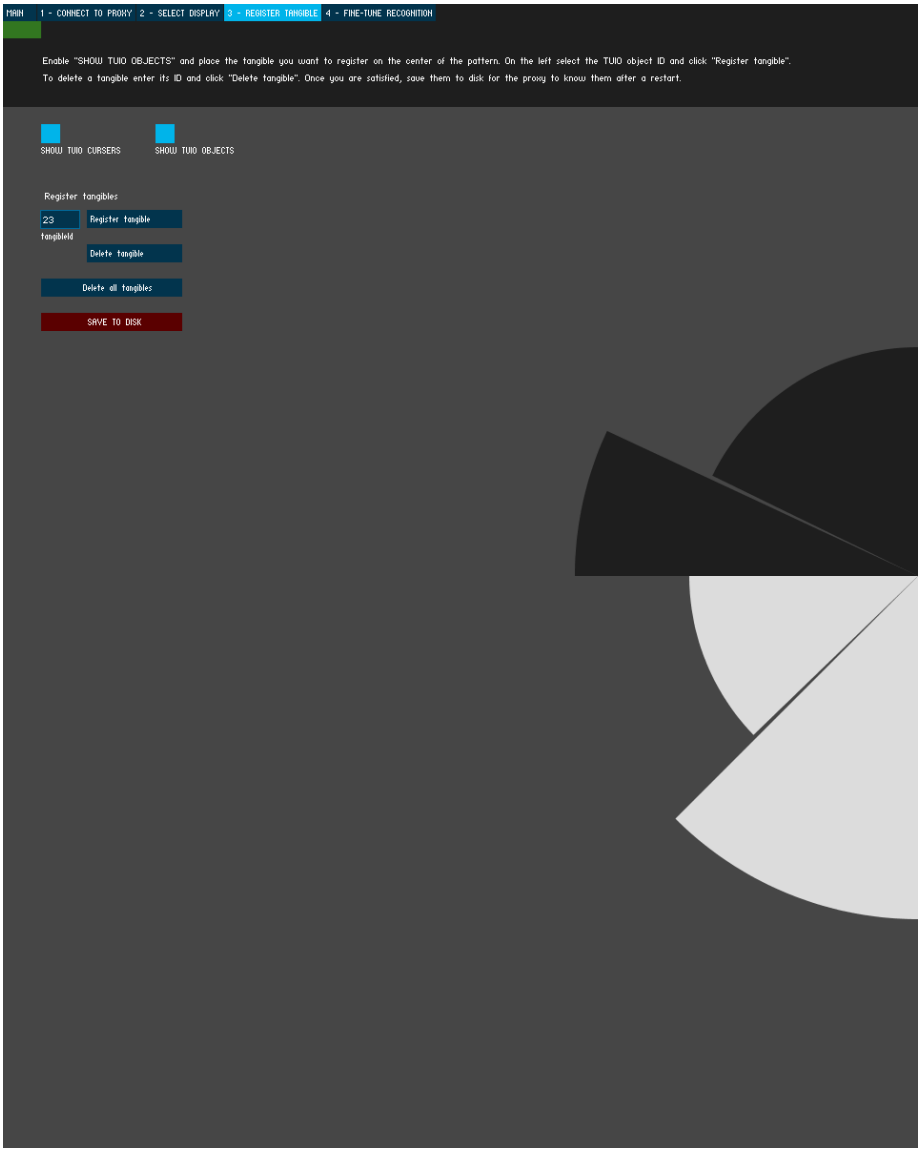
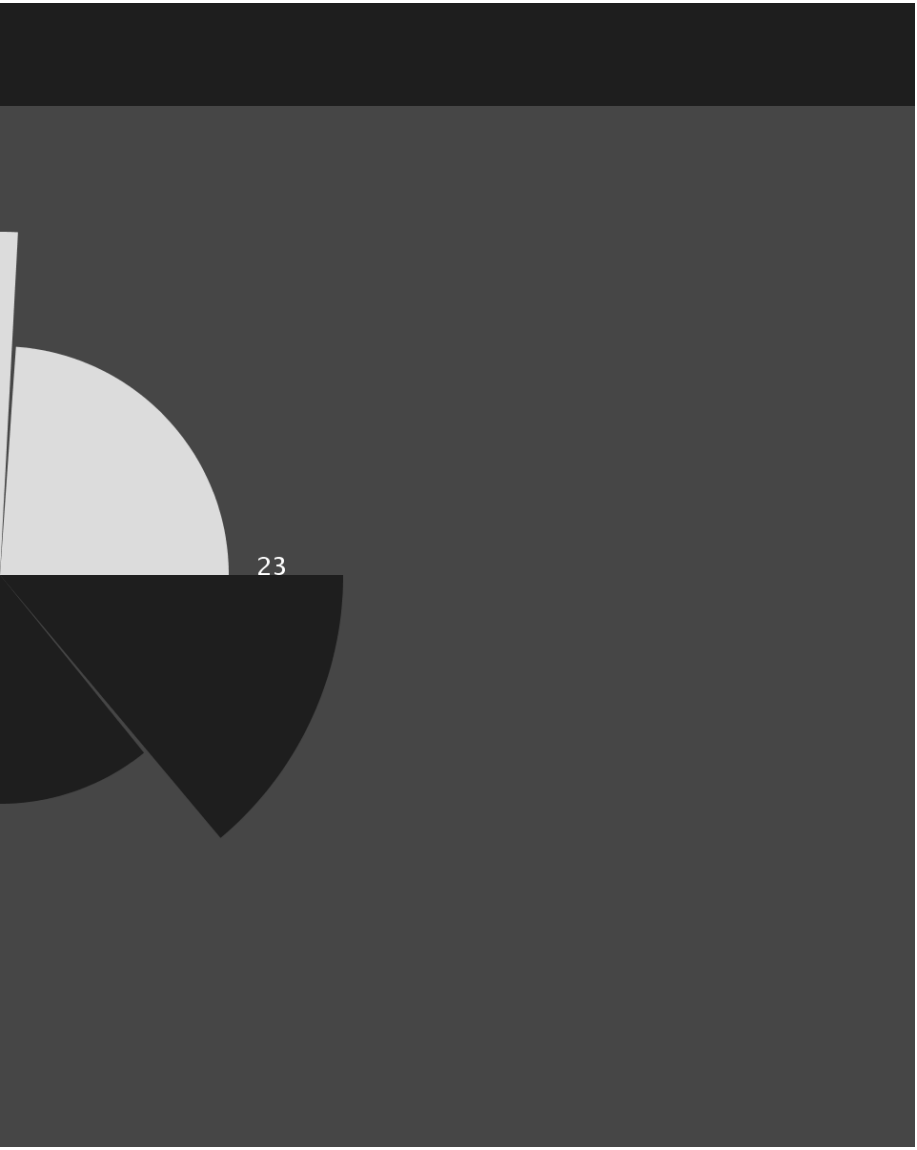


Figure 54. The tangible registration tab with a registered tangible


23

Fine tune recognition

The recognition might not be flawless yet and so the next tab lets you fine tune the proxy's configuration.

You can choose to filter all cursors that belong to the tangible and are not actual touches of real fingers.

As the user might grab the tangibles in a way that his fingers are recognized by the screen as touches you can activate a filter that removes all cursors in a circle around your tangible center. You can set the size of this circle with a slider.

There is also a simple low pass filter that prevents unusual rotations that are caused by a false recognition. There is also a slider to set the allowed rotation in radian in a refresh cycle. You can select if the rotation is filtered before or after the calculation of the tangible position. The "post" = after option is recommended.

If you feel like implementing a position filter you can do this in the python file filter.py and activate it by using the "filter position" button.

Keep in mind that every activated filter increases the latency of the system. Test the settings also in the user software, to decide if the filters are needed.

Furthermore can the essential parameters of the tangible recognition be changed.

The "tolerance" describes how much a tangible pattern can differ from it's original state when it was registered for the first time. Cursor positions might change a bit when other cursors are close to it. If the tolerance is too low the tangibles are not recognized anymore.

The needed votes setting sets how good a tangible has to be detected to be sure it is actually the right one. If you want to understand it in detail, read the part where the detection algorithm is described. If the tangible consists only of three points, this needs to be set to “1”.

To get feeling how good the tangible is recognized enable the “print votes...” option. With this option turned on, all the votes are printed in the console of the tangible.

One last setting, which is really important is the aspect ratio. If this is not set correct, the recognition fails as soon as a tangible is rotated.

If you want to keep your settings, save them to disk.

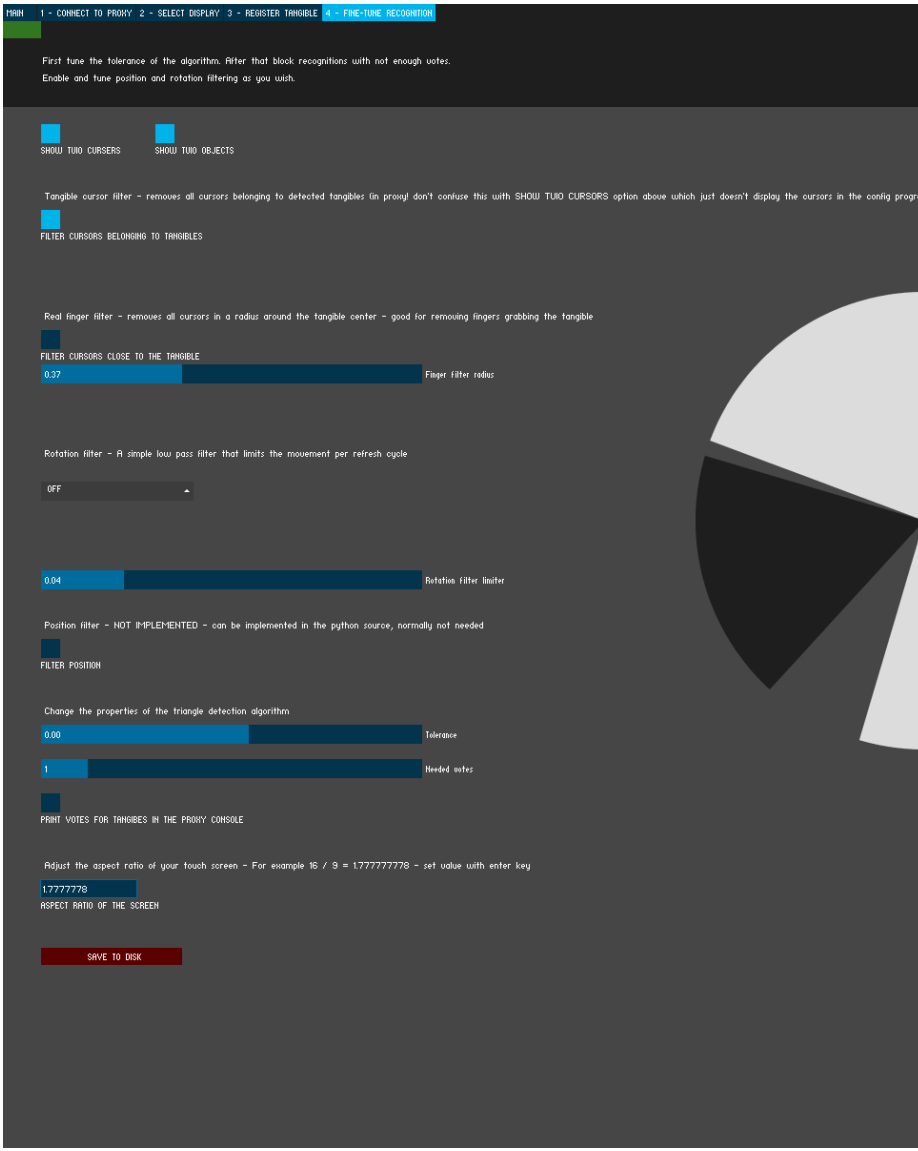
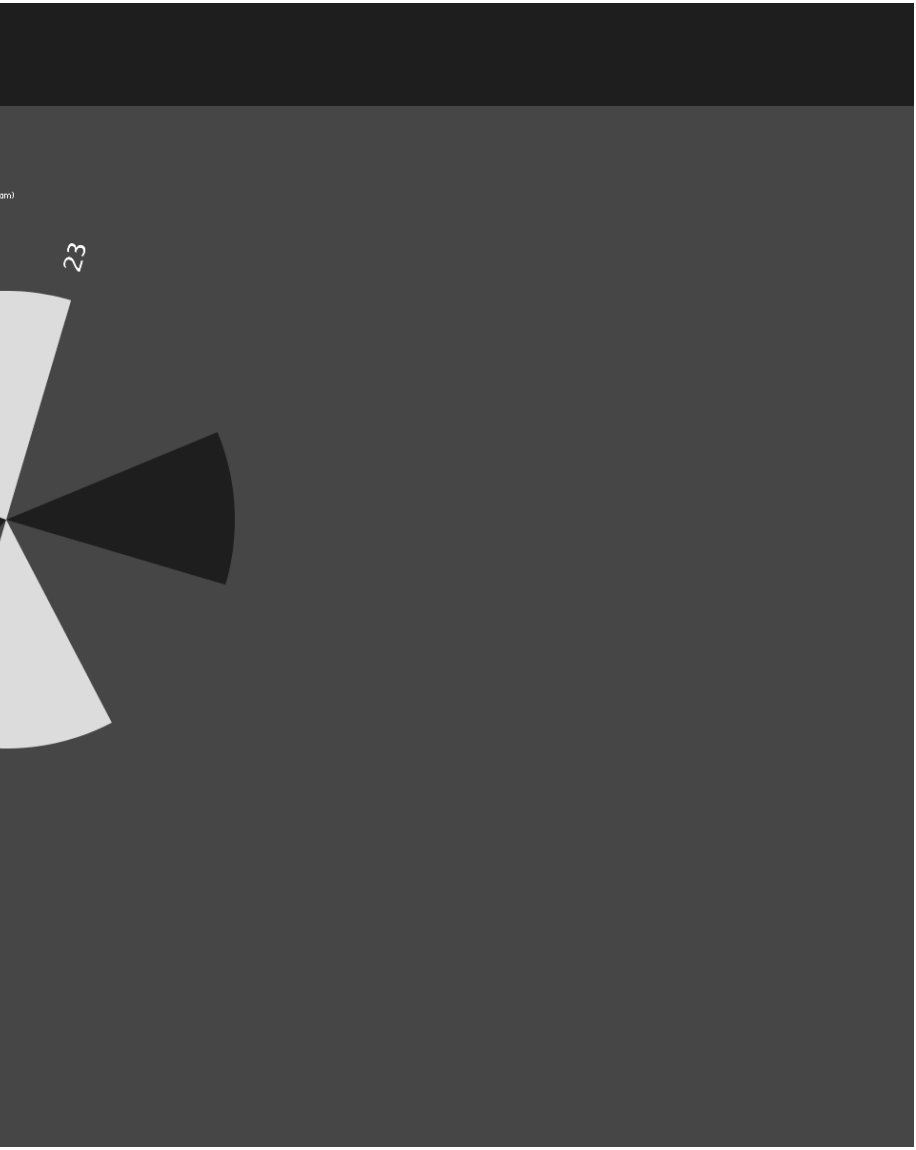


Figure 55. The fine tuning tab


am)
23

Important

As long as the GUI is running, no other program can receive the TUIO signals!

7. Usability test

7.1. The test software

The software to test the functionality of the project consists of two elements.

The first one is the setup and calibration program for the tangibles. It can be used to judge recognition rates and reaction speed.

The second one is a test case software. This software should show up:

- Whether the use of the tangibles improves the experience of controlling the software
- If the use of the tangibles interferes with the normal multi-touch usability

Therefor a program is created in which the user can build protons and neutrons out of quarks.

The basic setup is quite simply: An empty shell can be filled by dragging quarks into it. If the quarks in the shell are in the right constellation the quarks start to interact with each other and the shell becomes alive.

This can be done with just with one finger.

The next stage is using the tangibles to control the quarks in their position and have touch sensitive regions in or outside of them. With those regions the user can select for example the type of quark or display additional information.

Now the tangibles and touch-interactions are both used at the same time.

7.2. Evaluation of tracking accuracy

To check the accuracy a tangible with 3 false fingers (21 mm in diameter) is registered. All filters are turned of.

The tangible is then placed on the screen on a grid with 10 cm distance in horizontal and vertical direction. The following image shows you the deviation of the real tangible center to the displayed center in millimeters in case it is recognized.

	5	3	5	2	3	1	3	3	3	3	5
	12	6	5	8	2	2	2	3	3	4	5
	6	6	5	3	3	0	2	2	1	2	2
	-	6	10	5	1	4	4	9	10	4	4
	5	6	5	10	4	4	4	10	7	5	7

Figure 56. Deviation of real tangible position to calculated in millimeters

7.3. Evaluation of tracking robustness

To test the tracking robustness several test have been realized.

The tangibles used for those tests have been the following:

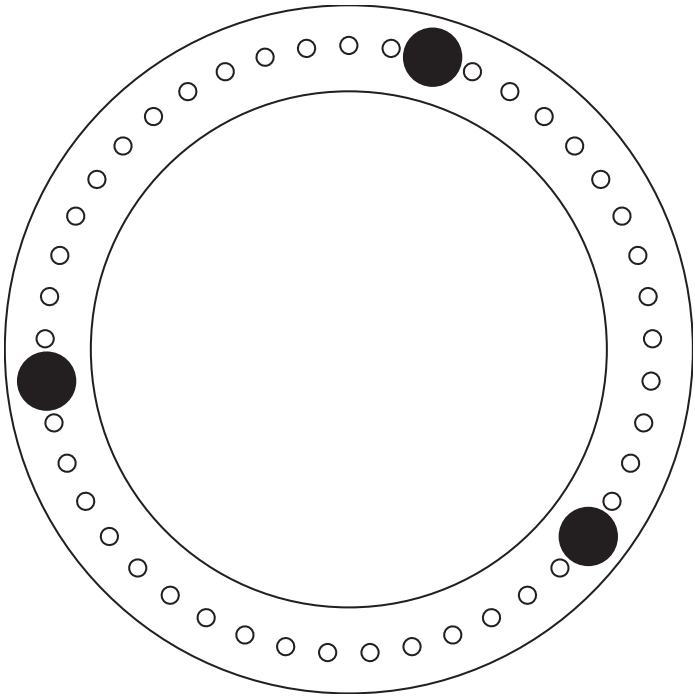


Figure 57. Tangible A3, outer diameter 120 mm

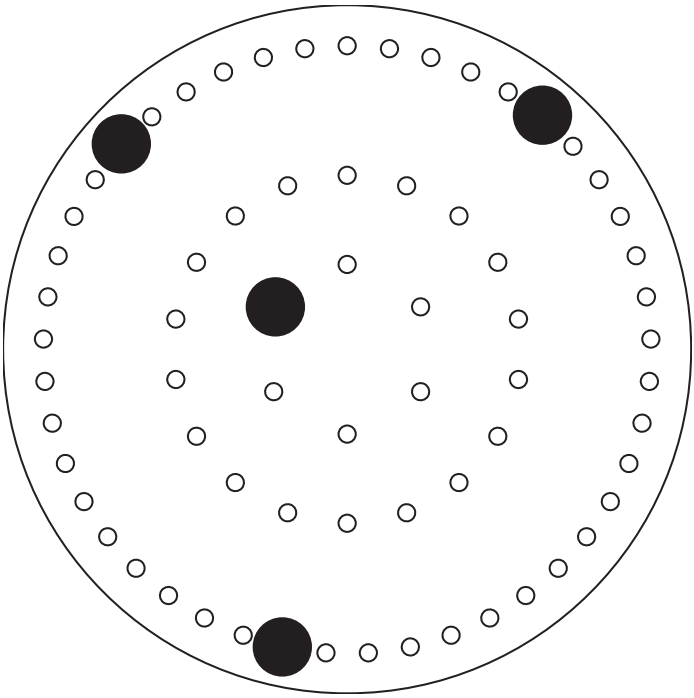


Figure 58. Tangible A4, outer diameter 120 mm

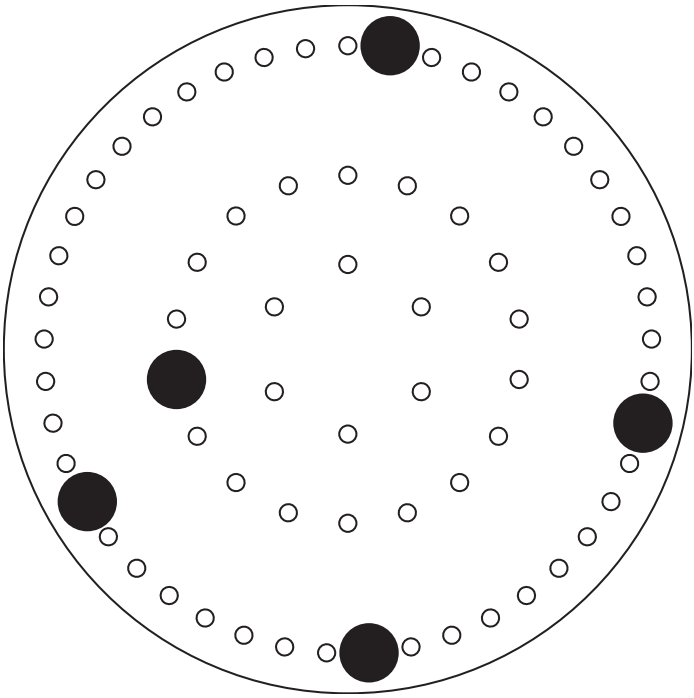


Figure 59. Tangible A5, outer diameter 120 mm

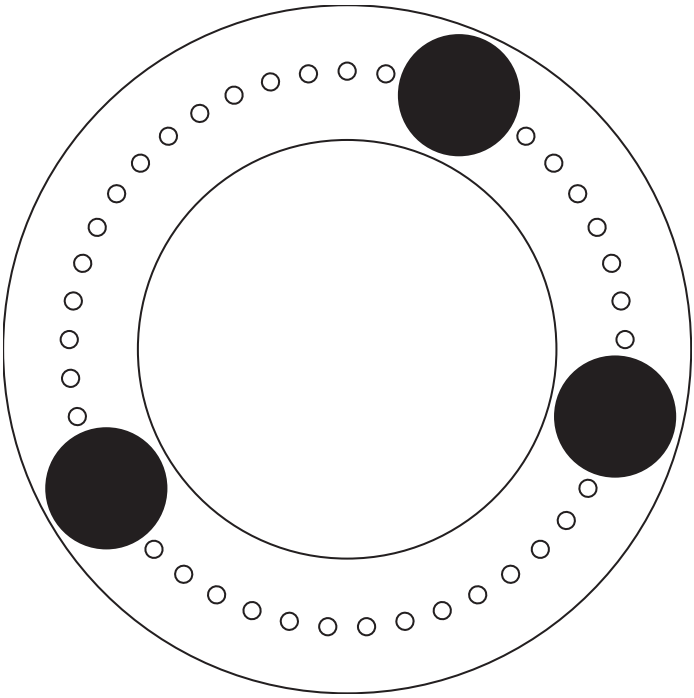


Figure 60. Tangible B3, outer diameter 120 mm

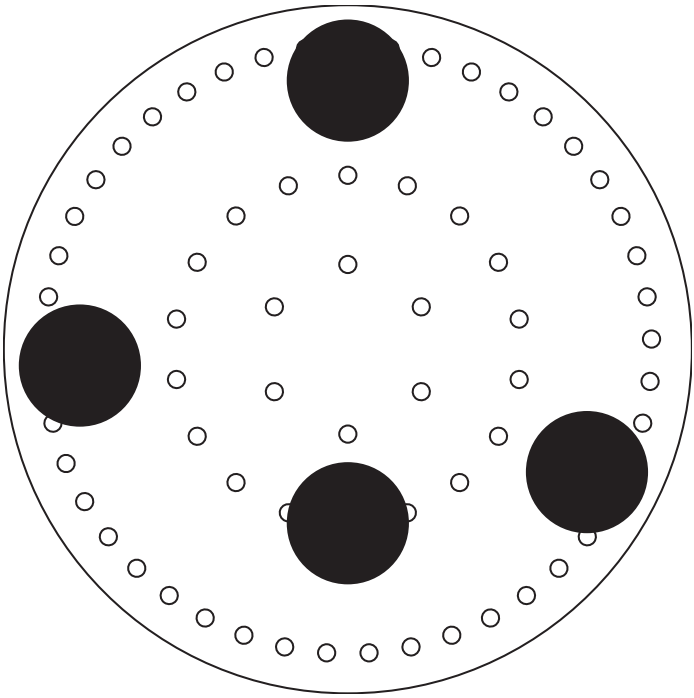


Figure 61. Tangible B4, outer diameter 120 mm

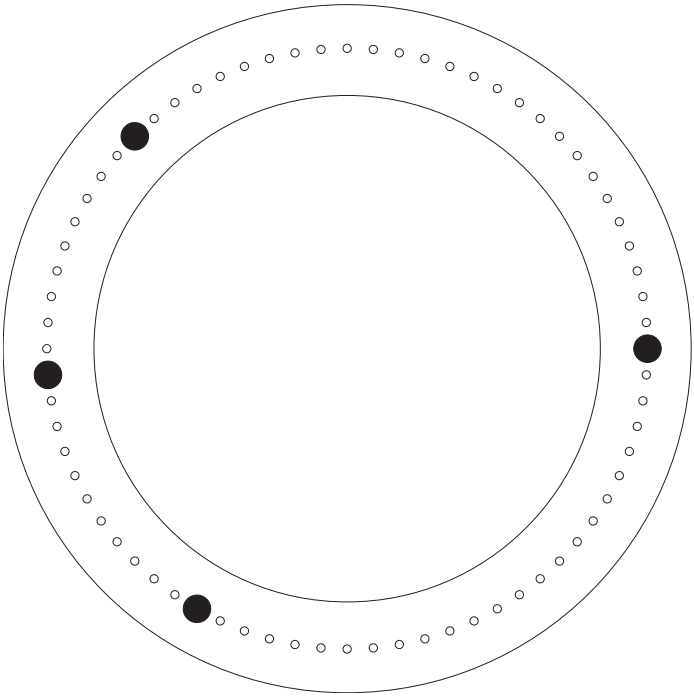


Figure 62. Tangible C4, outer diameter 250 mm

The drop test and mixed recognition

In this test the tangible is dropped on center of the screen and checked if it is recognized without further manual movement of the tangible.

Test with no other objects on screen:

Tangible	Filters	Needed votes	Tolerance	Recognized	
				Pos	Rot
A3	off	1	0.005	yes	yes
A4	off	1	0.005	yes	yes
A5	off	1	0.005	yes	yes
B3	off	1	0.005	yes	yes
B4	off	1	0.005	yes	yes
C4	off	1	0.005	yes	yes

Test with two 3 other tangibles on screen. All of them are registered and placed in the following positions:

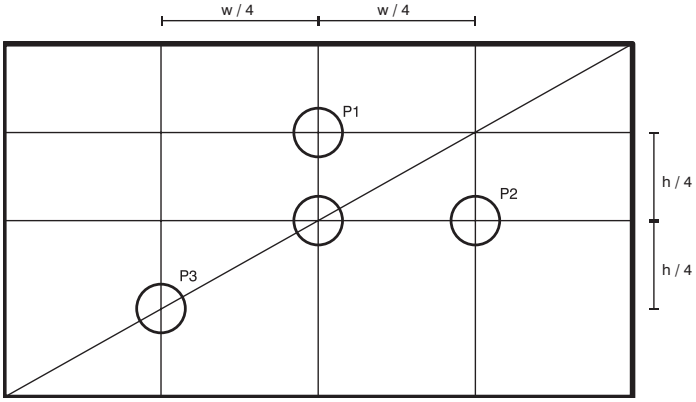


Figure 63. Scheme of placement

Tangible				Votes	Tol.	Recog.		Affecting position and rotation of other tangibles
	P1	P2	P3			Pos	Rot	
A3	A4	A5	B3	1	0.003	yes	yes	slightly
A4	B4	A5	B3	2	0.003	yes	yes	slightly
A5	A4	B4	B3	2	0.003	yes	yes	slightly
B3	B4	A4	A5	1	0.005	yes	yes	slightly
B4	A5	A4	B3	2	0.005	yes	yes	flickering
C4	A4	B4	A5	3	0.005	part	part	flickering A5

The 10 finger test

A tangible is placed on the screen center and recognized. 5 fingers are then placed on the right and left side of the tangible. All fingers are placed spread on the screen.

It is checked whether the recognition partially (flickering) or totally fails.

First is the tolerance set to 0. Then the tangible is placed on the screen. The tolerance is increased until the tangible is recognized. After this the fingers are placed on the screen. Even though other tangibles might me recognized in the finger pattern, taken into account is

only the impact on the centered tangible.

Test with spread fingers, important is a low tolerance setting still leading to stable tangible. The more “needed votes” possible, the better

Tangible	Filters	Needed votes	Tolerance	Flickering
				Pos
A3	off	1	0.004	seldom
A4	off	2	0.005	seldom
A5	off	6	0.0035	no
B3	off	1	0.0047	yes
B4	off	2	0.0034	yes
C4	off	3	0.0024	no

Swipe test

The tangible is swiped from the bottom left corner over the middle to the bottom right corner of the screen, with the settings from the “10 finger test”.

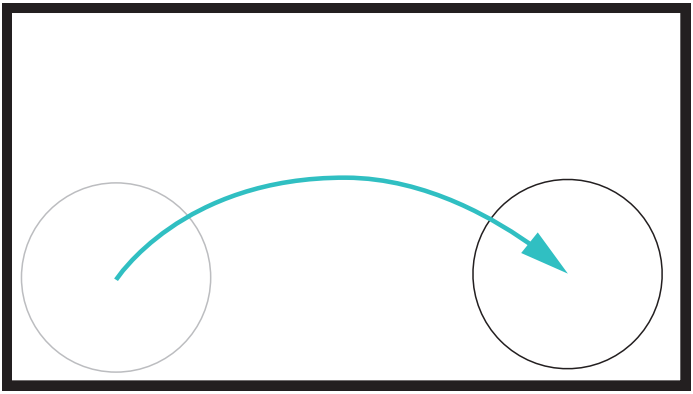


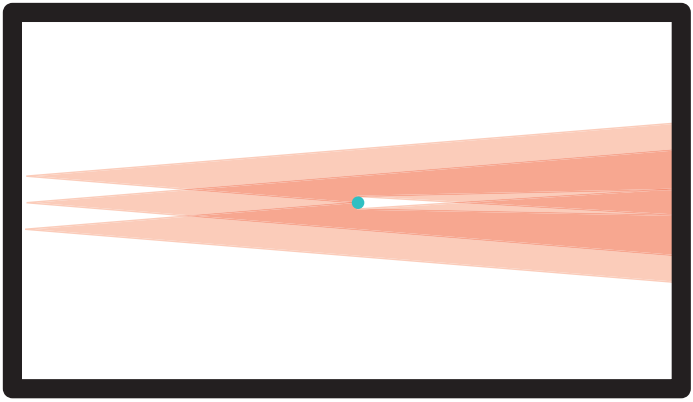
Figure 64. Motion for the swipe test

It is checked whether the recognition partially (flickering) or totally fails.

Tangible	Filters	Needed votes	Tolerance	Flickering
				Pos
A3	off	1	0.004	seldom
A4	off	2	0.005	seldom
A5	off	6	0.0035	no
B3	off	1	0.0047	yes
B4	off	2	0.0034	yes
C4	off	3	0.0024	no

7.4. Occlusion

The used hardware is more insensitive towards occlusion then one would think. This mainly due to the fact that the overlay doesn't work like an array of light barriers as described before.



Diodes sending light nonparallel.

Disturbances that occlude an object from one side, or even from two (if they do it on one axis) won't prevent the objects from being recognized, because there is still the other axis that can detect positions.

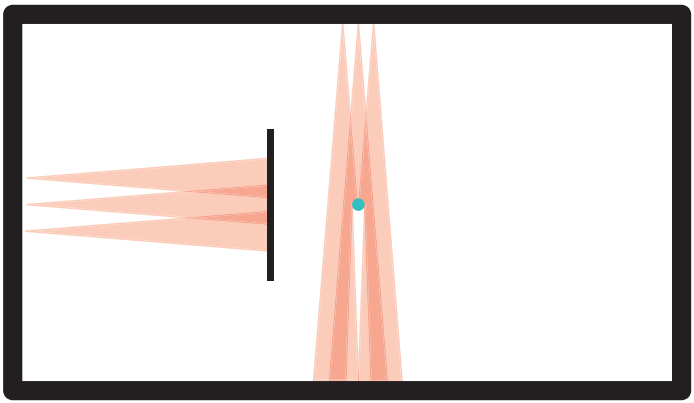


Figure 65. Occlusion of cursor from the side. The position can still be calculated.

But if the object is shielded on the vertical and horizontal axis it is not detectable anymore.

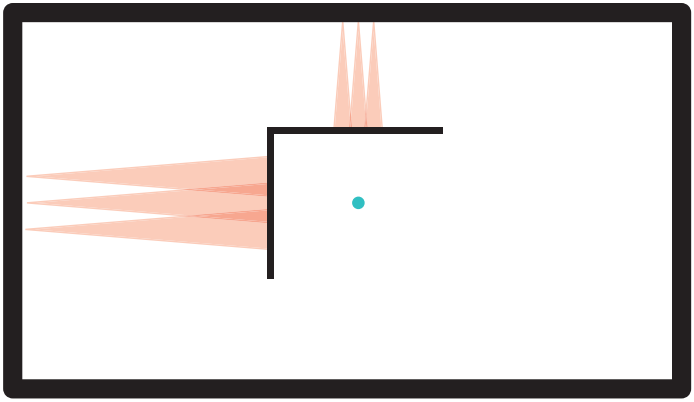


Figure 66. Object shielded on both, vertical and horizontal, axis.

Most of the time the fingers / false fingers / objects are not shielded by big barriers, there are just some minor objects disturbing the signal.

Even tough they are small objects they change the intensity of light received by the sensors. When two objects get close to each other the accuracy of their position goes down. The detected position changes, it is fuzzy.

The accuracy of the detection is better when both axis are in use. While shielding one complete axis during tests, it was quite obvious that bigger (20 mm) “false fingers” were detected better than small ones (10 mm). A lot of times the small objects weren’t detected at all.

If your tangible has a big size you should prefer the slightly bigger “false fingers” to have a more stable position.

But on a CD sized tangible (12 cm) this would make less sense, because of the close position of the objects towards each other. They would occlude each other.

In any case every difference of the detected position and the real position affects the recognition of the pattern. Fuzziness leads to the algorithm not finding the tangible, because the side lengths are out of the tolerances.

This is where the backup function steps in. It compares the alive cursors with the cursors that have been recognized as belonging to the tangibles before. Those that have been recognized are considered for a position calculation. See the algorithm chapter for more details.

As the change in position due to slight occlusion normally just changes by at most 1 cm on a 52 inch screen, the accuracy of the tangible position is still quite good.

7.5. Latency

The intermediate step of checking the cursors for patterns prolongs the latency of the whole signal chain.

The signal runtime from a touch to the visual representation does of course depend on a lot of factors:

- Overlay hardware
- Operating system
- Parser configuration
- Host hardware (CPU, memory, other processes)
- Python version (or binary optimization)
- The client software and it's performance

Due to all this different factors a measurement over the total runtime from signal to result would be interesting but not meaningful.

But it still is interesting to see how much time the calculation consumes on a sample system.

How the latency is measured

The sample system is a PC with

- Microsoft Windows 7 Enterprise
- Intel i7 3.2 GHz
- 12 GB RAM

what frankly doesn't say anything about the speed of this system. Just to get an idea.

The period measured starts when

- a TUIO "alive" message is received

and ends when

- all new TUIO cursors and objects belonging to this bundle are sent

To stop this time the python function time.clock() is used, as it allows the most precise measurement for python in windows. Values can be obtained in the range of microseconds. We are interested in the range of milliseconds.

Different modes to measure

The time needed for the calculations differs depending on what features are enabled:

- The amount of cursors
- If the cursors are checked for tangibles at all
- Check for tangibles without filtering of the cursors belonging to tangibles
- Check for tangibles with filtering of the cursors belonging to tangibles

- Check with or without position and rotation filtering

The measurements

For measurements one 3 point tangible is used. The other cursors are added in random positions.

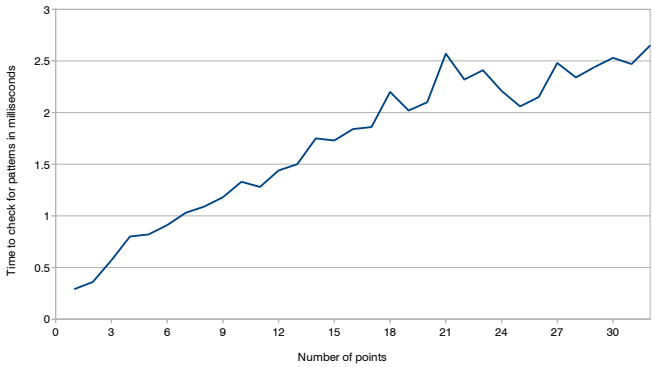


Figure 67. Comparison processing time to number of cursors without check for tangibles.

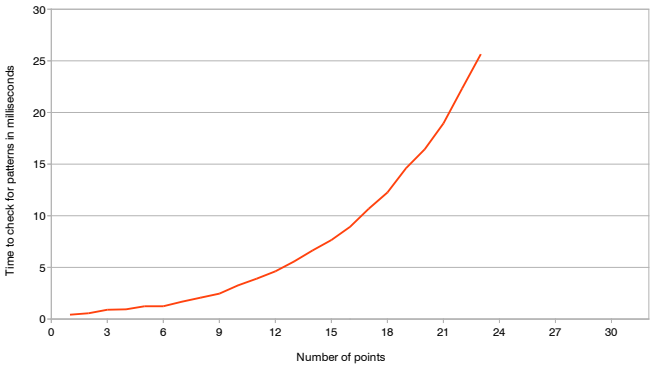


Figure 68. Comparison processing time to number of cursors with check for tangibles.

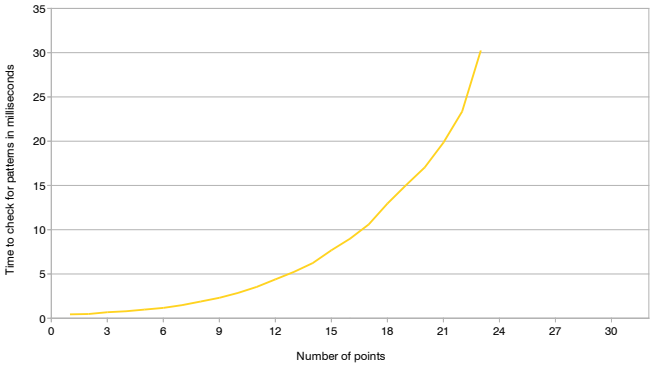


Figure 69. Comparison processing time to number of cursors with check for tangibles and filtering the tangible cursors that belong to tangibles and filtering rotation.

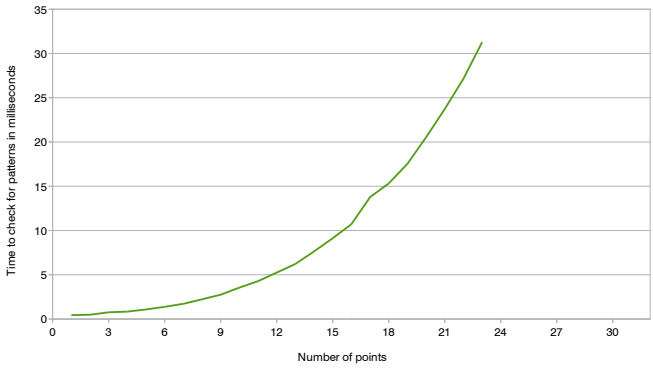


Figure 70. Comparison processing time to number of cursors with check for tangibles and filtering the tangible cursors that belong to tangibles and filtering rotation, filtering tangible touching fingers (2).

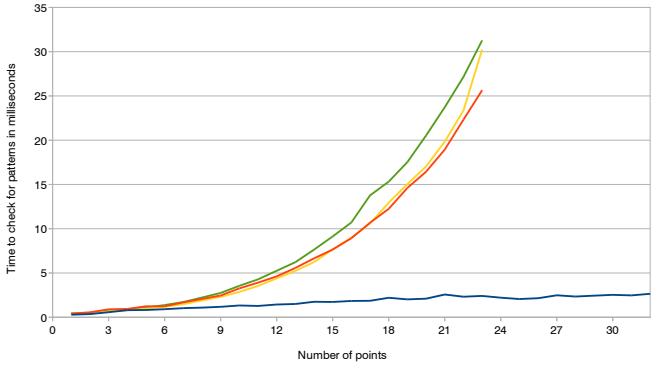


Figure 71. Comparison of latencies

7.6. The proton / neutron builder

The first step, using the tangibles to move the particles in the particle simulation software, functions in a satisfying way. The screen objects follow the movement of the tangibles.

The next step, to use also normal touch gestures with the software works, too. With some minor difficulties.

When the tangible cursors are filtered, the tangible can be moved over any reactive surface without triggering it.

If all the cursors in a certain distance around the tangible center are filtered, any use of touch sensitive areas in the circle is unfeasible. This includes buttons in the middle of ring shaped tangibles.

In case of non-circular filtering of the tangible cursors, which enables the ability to use buttons in the tangible area, another problem exists: The fingers grabbing the tangible are likely to trigger buttons and other touch reactive areas.

One solution to solve those two problems would be a filtering of cursors in a ring shaped filter area or in even more complex shapes.

With the use of more complex shapes the use of the elbow shaped tangible could gain in attractiveness, because a circular area filter renders a large area around the tangible unusable for touch interactions.

Figure 72. Circular filtering around elbow. The red area is not usable for touch interactions.

Further a problem occurrs when the tangibles are used in a more active way in terms of throwing them on the screen or lift them from it. As long as the tangibles are not yet recognized for the first time, their cursors are still considered to be human fingers. They are not filtered and can trigger buttons.

This also occurs when the tangibles are lifted of the surface. This mostly happens when tangibles are “wildly” manoeuvred over the glass surface and lifted on one side.

7.7. Summary of difficulties

The use of multiple tangibles is still not satisfying. This has several reasons.

- One is that the hardware is not very accurate. Shifts of recognized positions are the biggest problem. Once multiple tangibles are used they start to shield each other and disturb the recognition which leads to shifted cursor positions.

- By using tangibles with a lot of false fingers they maximum number of recognizable points is soon reached.

- The height of the false fingers is one of the reasons that make the tangible appear bulky. A lower position of the LEDs would improve the appearance, but then again the tangible itself would be quite likely to be recognized as object.

- A “wild and rigorous” use is still buggy. This is important as the system should be able to cope situations that appear in when children interact with it.

- The filtering of tangible cursors needs improvement, this should be done when the code is transferred into C++, due to more capabilities in terms of performance.

8. Related work

8.1. TUIC

Related to this work is TUIC. [Yu, N.-H. & Chan,L.-W ... 2011] It is created for capacitive touch screens.

This system combines two ideas:

- Spatial recognition of tangibles
- Frequency recognition

Spatial recognition

Each point has a passive circuit to be recognized by the screen.

The TUIC-2D approach has 3 registration points which are known in their angle and distance towards each other.

If 3 points appear in such a constellation, it is likely that they belong to a tangible. Position and rotation of the tangible can be calculated.

One, or more, payload point is added in addition to the registration points. The payload points in predefined positions can form a tangible ID to distinguish the tangibles from each other.

Frequency recognition

To reduce the number of used points the payload points are replaced by a single one. This point is connected to an active circuit which switches the point on and off.

The state switching simulates rapid touches in the magnitude of 15 ms and more. The different tangibles are

distinguished by their frequency. For example tangible one's cycle is "on" for 15 ms and 15 ms "off". For tangible two it is 20 ms each.

Hybrid of spatial and frequency recognition

The combined use of both approaches leads to a three point tangible. One frequency point to provide the tangible ID, two passive points two allow the calculation of orientation.

Relation to this thesis

The payload encoding can also be used with the infrared overlay, but is not likely to work in a satisfying way because of the IR grid. A bulk of points is likely to be recognized as one single touch or object. The grid can't "see" the points that are covered by others.

Even though the approach in this thesis and TUIC-2D share the idea of encoding the tangible ID in the position of touch points.

The frequency encoding would require optical solution which would result in a quite complicated fiducial with high production costs compared to a completely passive tangible.

8.2. TouchPlanVS Lite

TouchPlanVS [Brosda, C. & Daemen, J. & Djuderija, S. & Joeres, S. ... (2012)] is a software to plan and control scene flow and settings in virtual studios with the help of tangibles on capacitive touch screens developed at the FH Düsseldorf.

Capacitive tangibles are placed on iPads to control features in the software. The tangibles simulate finger touches by having three contact points which are connected to the fingers grabbing the tangible through conductive material.

Relation to this thesis

TouchPlanVS Lite contains also an algorithm to detect triangles which is similar to the TUIO proxy alogrithm in that point that it compares triangle side lengths and is also based on Groth's work.

The TUIO proxy can also be used, without any modifications, to detect capacitve tangibles that simulate fingertouches, as it works with every TUIO signal.

9. Future prospects

A forecast on the meaningfulness of this thesis in 5 years of time...

Those forecasts have always been wrong, predicting the future is a hard business and so far everybody failed more or less.

This approach is meant as a low cost fix for a situation that exists now. It has been done because of the failure or flaws of other systems at the moment. From the first concept on it has been thought of as a gap filler and it still is.

I hope that pretty soon techniques like the Microsoft PixelSense will improve, at least the idea behind it seems promising to me.

I hope there will be a cheaper and more accurate solution to object recognition than this in 5 years of time.

Conclusion

Insights gained through this thesis

In my opinion this system can help to use one or two tangibles on a infrared overlay in a really satisfying way.

The use of more tangibles is possible, but depends on the quality of recognition of the used hardware. With the tested hardware the use of e.g. 4 tangibles is not flawless.

In terms of latency it is obvious that Python isn't the best choice in terms of performance. Python was chosen for rapid development in this proof-of-concept phase. A port to C++ is expected to decrease the latency of the proxy in most cases down to at least less than a couple of milliseconds.

This project offered a challenge to me and a chance to demonstrate the engineering skills I have acquired during my studies at the University of Applied Sciences Düsseldorf and my internship at CERN. I find the combination of software and hardware engineering as applied in tangible computing to be the perfect synthesis of what I have learned and hope to continue to work and research in the field.

References

Literature

Arzoumanian, K.,Holmberg,J. & Norman,B. (2005)
"An astronomical pattern-matching algorithm for computer-aided identification of whale sharks Rhincodon typus."
Journal of Applied Ecology, 42, 999-1011.

Brosda, C. & Daemen, J. & Djuderija, S. & Joeres, S. & Langer, O. & Schweitzer, A. & Wilhelm, A. & Herder, J. (2012)
TouchPlanVS Lite – A Tablet-based Tangible Multitouch Planning System for Virtual TV Studio Productions
The Joint International Conference on Human-Centered Computer Environments, Hamamatsu/Aizu-Wakamatsu/ Duesseldorf

Groth, E.J. (1986)
A pattern-matching algorithm for two-dimensional coordinate lists.
Astronomical Journal, 91, 1244-1248.

Kaltenbrunner, M. & Bovermann, T. & Bencina, R. & Costanza, E. (2005)
"TUIO - A Protocol for Table-Top Tangible User Interfaces"
Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation, Vannes (France)

Kaltenbrunner, M. & Bencina, R. (2007)
reacTIVision: A Computer-Vision Framework for Table-Based Tangible Interaction
Proceedings of the first international conference on "Tangible and Embedded Interaction" (TEI07). Baton Rouge, Louisiana

Microsoft PixelSense
http://www.microsoft.com/en-us/pixelsense/pixelsense.aspx
Accessed June, 21st 2012

Processing
http://processing.org/
Accessed August, 2nd 2012

PQ Labs
http://multi-touch-screen.com/product_g3.html
Accessed August, 2nd 2012

tuio.org
http://www.tuio.org/?specification
Accessed July, 6th 2012

US Patent Aplication 20120098753
http://www.faqs.org/patents/app/20120098753
Accessed August, 3rd 2012

Wikipedia Touchscreen
http://en.wikipedia.org/wiki/Touchscreen#Capacitive
Accessed June, 21st 2012

Wright, M., Freed, A., Momeni A. (2003)
"OpenSound Control: State of the Art 2003"
Proceedings of the 3rd Conference on New Instruments for Musical Expression (NIME 03), Montreal, Canada.

Yu,N.-H & Chan, L.-W. & Lau, S.-Y. & Tsai, S.-S. & Hsiao, I-C. & Tsai, D.-J. & Cheng, L.-P. & Hsiao, F.-I & Chen, M. Y. & Huang, P. & Hung, Y.-P. (2011)
TUIC: Enabling Tangible Interaction on Capacitive Multi-touch Displays
ACM SIGCHI

Figures

All figures, except the logos of the “Virtual Sets and Virtual Environments Laboratory” of the FH Düsseldorf, UAS and the European Organization for Nuclear Research on the cover, have been created by the author.

Figure 73. Contents

A. Software installation

The following steps describe the installation on Windows 7:

Install Python 2.7

- Install python 2.7 from http://www.python.org/getit/

Don't forget to add your python directory to the "Path" environment variable in Advanced System Settings!

Install pyOSC

- Download zip from https://trac.v2.nl/wiki/pyOSC
- Unpack file
- Open a command line and go to the folder with the extracted files
- Run 'setup.py install'

Install Processing

- Follow instructions on http://processing.org/learning/gettingstarted/

Install controlP5, oscP5, netP5,

- Download and install controlP5 from http://www.sojamo.de/libraries/controlP5/

- Download and install oscP5 from http://www.sojamo.de/libraries/oscP5/

Install FullScreen API For Processing

- Install from http://www.superduper.org/processing/fullscreen_api/

PQ-Labs parser modification

The PQ-Labs software for the infrared overlay should already be installed by you.

The file “Program Files\PQLabs\MultiTouchPlatform\mtsvrset.xml” has to be altered in order to reroute all the TUIO messages to the proxy.

The UDP port has to be changed from 3333 to 3332.

It should look like this:

```
<tuio tuio_support=”true” flash_tuio_support=”true”>

   <server type=”udp” host=”127.0.0.1” port=”3332” />

   <server type=”tcp” host=”127.0.0.1” port=”3000” />

</tuio>
```

B. Open source publishing

In my opinion patenting and restricting software, medication and other thoughts or things is slowing down the evolution of our society down.

The TUIO proxy and the control software are published under the GPLv3.

The latest code is downloadable on github:

https://github.com/hoshijirushi/Vega

GNU GENERAL PUBLIC LICENSE Version 3

29 June 2007

Preamble

The GNU General Public License is a free, copyleft license for software and other kinds of works.

The licenses for most software and other practical works are designed to take away your freedom to share and change the works. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change all versions of a program--to make sure it remains free software for all its users. We, the Free Software Foundation, use the GNU General Public License for most of our software; it applies also to any other work released this way by its authors. You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for them if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs, and that you know you can do these things.

To protect your rights, we need to prevent others from denying you these rights or asking you to surrender the rights. Therefore, you have certain responsibilities if you distribute copies of the software, or if you modify it: responsibilities to respect the freedom of others.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must pass on to the recipients the same freedoms that you received. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

Developers that use the GNU GPL protect your rights with two steps:

(1) assert copyright on the software, and (2) offer you this License giving you legal permission to copy, distribute and/or modify it.

For the developers' and authors' protection, the GPL clearly explains that there is no warranty for this free software. For both users' and authors' sake, the GPL requires that modified versions be marked as changed, so that their problems will not be attributed erroneously to authors of previous versions.

Some devices are designed to deny users access to install or run modified versions of the software inside them, although the manufacturer can do so. This is fundamentally incompatible with the aim of protecting users' freedom to change the software. The systematic pattern of such abuse occurs in the area of products for individuals to use, which is precisely where it is most unacceptable. Therefore, we have designed this version of the GPL to prohibit the practice for those products. If such problems arise substantially in other domains, we stand ready to

extend this provision to those domains in future versions of the GPL, as needed to protect the freedom of users.

Finally, every program is threatened constantly by software patents. States should not allow patents to restrict development and use of software on general-purpose computers, but in those that do, we wish to avoid the special danger that patents applied to a free program could make it effectively proprietary. To prevent this, the GPL assures that patents cannot be used to render the program non-free.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS

0. Definitions.

“This License” refers to version 3 of the GNU General Public License.

“Copyright” also means copyright-like laws that apply to other kinds of works, such as semiconductor masks.

“The Program” refers to any copyrightable work licensed under this License. Each licensee is addressed as “you”. “Licensees” and “recipients” may be individuals or organizations.

To “modify” a work means to copy from or adapt all or part of the work in a fashion requiring copyright permission, other than the making of an exact copy. The resulting work is called a “modified version” of the earlier work or a work “based on” the earlier work.

A “covered work” means either the unmodified Program or a work based on the Program.

To “propagate” a work means to do anything with it that, without permission, would make you directly or secondarily liable for infringement under applicable copyright law, except executing it on a computer or modifying a private copy. Propagation includes copying, distribution (with or without modification), making available to the public, and in some countries other activities as well.

To “convey” a work means any kind of propagation that enables other parties to make or receive copies. Mere interaction with a user through a computer network, with no transfer of a copy, is not conveying.

An interactive user interface displays “Appropriate Legal Notices” to the extent that it includes a convenient and prominently visible feature that (1) displays an appropriate copyright notice, and (2) tells the user that there is no warranty for the work (except to the extent that warranties are provided), that licensees may convey the work under this License, and how to view a copy of this License. If the interface presents a list of user commands or options, such as a menu, a prominent item in the list meets this criterion.

1. Source Code.

The “source code” for a work means the preferred form of the work for making modifications to it. “Object code” means any non-source form of a work.

A “Standard Interface” means an interface that either is an official standard defined by a recognized standards body, or, in the case of interfaces specified for a particular programming language, one that is widely used among developers working in that language.

The “System Libraries” of an executable work include anything, other than the work as a whole, that (a) is included in the normal form of packaging a Major Component, but which is not part of that Major Component, and (b) serves only to enable use of the work with that Major Component, or to implement a Standard Interface for which an implementation is available to the public in source code form. A “Major Component”, in this context, means a major essential component (kernel, window system, and so on) of the specific operating system (if any) on which the executable work runs, or a compiler used to produce the work, or an object code interpreter used to run it.

The “Corresponding Source” for a work in object code form means all the source code needed to generate, install, and (for an executable work) run the object code and to modify the work, including scripts to control those activities. However, it does not include the work’s System Libraries, or general-purpose tools or generally available free programs which are used unmodified in performing those activities but which are not part of the work. For example, Corresponding Source includes interface definition files associated with source files for the work, and the source code for shared libraries and dynamically linked subprograms that the work is specifically designed to require, such as by intimate data communication or control flow between those subprograms and other parts of the work.

The Corresponding Source need not include anything that users can regenerate automatically from other parts of the Corresponding Source.

The Corresponding Source for a work in source code form is that same work.

2. Basic Permissions.

All rights granted under this License are granted for the term of copyright on the Program, and are irrevocable provided the stated conditions are met. This License explicitly affirms your unlimited permission to run the unmodified Program. The output from running a covered work is covered by this License only if the output, given its content, constitutes a covered work. This License acknowledges your rights of fair use or other equivalent, as provided by copyright law.

You may make, run and propagate covered works that you do not convey, without conditions so long as your license otherwise remains in force. You may convey covered works to others for the sole purpose of having them make modifications exclusively for you, or provide you with facilities for running those works, provided that you comply with the terms of this License in conveying all material for which you do not control copyright. Those thus making or running the covered works for you must do so exclusively on your behalf, under your direction and control, on terms that prohibit them from making any copies of your copyrighted material outside their relationship with you.

Conveying under any other circumstances is permitted solely under the conditions stated below. Sublicensing is not allowed; section 10 makes it unnecessary.

3. Protecting Users' Legal Rights From Anti-Circumvention Law.

No covered work shall be deemed part of an effective technological measure under any applicable law fulfilling obligations under article 11 of the WIPO copyright treaty

adopted on 20 December 1996, or similar laws prohibiting or restricting circumvention of such measures.

When you convey a covered work, you waive any legal power to forbid circumvention of technological measures to the extent such circumvention is effected by exercising rights under this License with respect to the covered work, and you disclaim any intention to limit operation or modification of the work as a means of enforcing, against the work's users, your or third parties' legal rights to forbid circumvention of technological measures.

4. Conveying Verbatim Copies.

You may convey verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice; keep intact all notices stating that this License and any non-permissive terms added in accord with section 7 apply to the code; keep intact all notices of the absence of any warranty; and give all recipients a copy of this License along with the Program.

You may charge any price or no price for each copy that you convey, and you may offer support or warranty protection for a fee.

5. Conveying Modified Source Versions.

You may convey a work based on the Program, or the modifications to produce it from the Program, in the form of source code under the terms of section 4, provided that you also meet all of these conditions:

a) The work must carry prominent notices stating that you modified it, and giving a relevant date.

b) The work must carry prominent notices stating that it is released under this License and any conditions added under section

7. This requirement modifies the requirement in section 4 to “keep intact all notices”.

c) You must license the entire work, as a whole, under this License to anyone who comes into possession of a copy. This License will therefore apply, along with any applicable section 7 additional terms, to the whole of the work, and all its parts, regardless of how they are packaged. This License gives no permission to license the work in any other way, but it does not invalidate such permission if you have separately received it.

d) If the work has interactive user interfaces, each must display Appropriate Legal Notices; however, if the Program has interactive interfaces that do not display Appropriate Legal Notices, your work need not make them do so.

A compilation of a covered work with other separate and independent works, which are not by their nature extensions of the covered work, and which are not combined with it such as to form a larger program, in or on a volume of a storage or distribution medium, is called an “aggregate” if the compilation and its resulting copyright are not used to limit the access or legal rights of the compilation’s users beyond what the individual works permit. Inclusion of a covered work in an aggregate does not cause this License to apply to the other parts of the aggregate.

6. Conveying Non-Source Forms.

You may convey a covered work in object code form under the terms of sections 4 and 5, provided that you also convey the machine-readable Corresponding Source under the terms of this License, in one of these ways:

a) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by the Corresponding Source fixed on a durable physical medium customarily used for software interchange.

b) Convey the object code in, or embodied in, a physical product (including a physical distribution medium), accompanied by a written offer, valid for at least three years and valid for as long as you offer spare parts or customer support for that product model, to give anyone who possesses the object code either (1) a copy of the Corresponding Source for all the software in the product that is covered by this License, on a durable physical medium customarily used for software interchange, for a price no more than your reasonable cost of physically performing this conveying of source, or (2) access to copy the Corresponding Source from a network server at no charge.

c) Convey individual copies of the object code with a copy of the written offer to provide the Corresponding Source. This alternative is allowed only occasionally and noncommercially, and only if you received the object code with such an offer, in accord with subsection 6b.

d) Convey the object code by offering access from a designated place (gratis or for a charge), and offer equivalent access to the Corresponding Source in the same way through the same place at no further charge. You need not require recipients to copy the Corresponding Source along with the object code. If the place to copy the object code is a network server, the Corresponding Source may be on a different server (operated by you or a third party) that supports equivalent copying facilities, provided you maintain clear directions next to the object code saying where to find the Corresponding Source. Regardless of what server hosts the Corresponding

Source, you remain obligated to ensure that it is available for as long as needed to satisfy these requirements.

e) Convey the object code using peer-to-peer transmission, provided you inform other peers where the object code and Corresponding Source of the work are being offered to the general public at no charge under subsection 6d.

A separable portion of the object code, whose source code is excluded from the Corresponding Source as a System Library, need not be included in conveying the object code work.

A “User Product” is either (1) a “consumer product”, which means any tangible personal property which is normally used for personal, family, or household purposes, or (2) anything designed or sold for incorporation into a dwelling. In determining whether a product is a consumer product, doubtful cases shall be resolved in favor of coverage. For a particular product received by a particular user, “normally used” refers to a typical or common use of that class of product, regardless of the status of the particular user or of the way in which the particular user actually uses, or expects or is expected to use, the product. A product is a consumer product regardless of whether the product has substantial commercial, industrial or non-consumer uses, unless such uses represent the only significant mode of use of the product.

“Installation Information” for a User Product means any methods, procedures, authorization keys, or other information required to install and execute modified versions of a covered work in that User Product from a modified version of its Corresponding Source. The information must suffice to ensure that the continued functioning of the modified object code is in no case prevented or interfered with solely because modification has been made.

If you convey an object code work under this section in, or with, or specifically for use in, a User Product, and the conveying occurs as part of a transaction in which the right of possession and use of the User Product is transferred to the recipient in perpetuity or for a fixed term (regardless of how the transaction is characterized), the Corresponding Source conveyed under this section must be accompanied by the Installation Information. But this requirement does not apply if neither you nor any third party retains the ability to install modified object code on the User Product (for example, the work has been installed in ROM).

The requirement to provide Installation Information does not include a requirement to continue to provide support service, warranty, or updates for a work that has been modified or installed by the recipient, or for the User Product in which it has been modified or installed. Access to a network may be denied when the modification itself materially and adversely affects the operation of the network or violates the rules and protocols for communication across the network.

Corresponding Source conveyed, and Installation Information provided, in accord with this section must be in a format that is publicly documented (and with an implementation available to the public in source code form), and must require no special password or key for unpacking, reading or copying.

7. Additional Terms.

“Additional permissions” are terms that supplement the terms of this License by making exceptions from one or more of its conditions. Additional permissions that are applicable to the entire Program shall be treated as though they were included in this License, to the extent that they are valid under applicable law. If additional permissions apply only to part of the Program, that part may be

used separately under those permissions, but the entire Program remains governed by this License without regard to the additional permissions.

When you convey a copy of a covered work, you may at your option remove any additional permissions from that copy, or from any part of it. (Additional permissions may be written to require their own removal in certain cases when you modify the work.) You may place additional permissions on material, added by you to a covered work, for which you have or can give appropriate copyright permission.

Notwithstanding any other provision of this License, for material you add to a covered work, you may (if authorized by the copyright holders of that material) supplement the terms of this License with terms:

a) Disclaiming warranty or limiting liability differently from the terms of sections 15 and 16 of this License; or

b) Requiring preservation of specified reasonable legal notices or author attributions in that material or in the Appropriate Legal Notices displayed by works containing it; or

c) Prohibiting misrepresentation of the origin of that material, or requiring that modified versions of such material be marked in reasonable ways as different from the original version; or

d) Limiting the use for publicity purposes of names of licensors or authors of the material; or

e) Declining to grant rights under trademark law for use of some trade names, trademarks, or service marks; or

f) Requiring indemnification of licensors and authors of that material by anyone who conveys the material (or modified versions of it) with contractual assumptions of liability to the recipient, for any liability that these contractual assumptions directly impose on those licensors and authors.

All other non-permissive additional terms are considered "further restrictions" within the meaning of section 10. If the Program as you received it, or any part of it, contains a notice stating that it is governed by this License along with a term that is a further restriction, you may remove that term. If a license document contains a further restriction but permits relicensing or conveying under this License, you may add to a covered work material governed by the terms of that license document, provided that the further restriction does not survive such relicensing or conveying.

If you add terms to a covered work in accord with this section, you must place, in the relevant source files, a statement of the additional terms that apply to those files, or a notice indicating where to find the applicable terms.

Additional terms, permissive or non-permissive, may be stated in the form of a separately written license, or stated as exceptions; the above requirements apply either way.

8. Termination.

You may not propagate or modify a covered work except as expressly provided under this License. Any attempt otherwise to propagate or modify it is void, and will automatically terminate your rights under this License (including any patent licenses granted under the third paragraph of section 11).

However, if you cease all violation of this License, then your license from a particular copyright holder is

reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, you do not qualify to receive new licenses for the same material under section 10.

9. Acceptance Not Required for Having Copies.

You are not required to accept this License in order to receive or run a copy of the Program. Ancillary propagation of a covered work occurring solely as a consequence of using peer-to-peer transmission to receive a copy likewise does not require acceptance. However, nothing other than this License grants you permission to propagate or modify any covered work. These actions infringe copyright if you do not accept this License. Therefore, by modifying or propagating a covered work, you indicate your acceptance of this License to do so.

10. Automatic Licensing of Downstream Recipients.

Each time you convey a covered work, the recipient automatically receives a license from the original licensors,

to run, modify and propagate that work, subject to this License. You are not responsible for enforcing compliance by third parties with this License.

An “entity transaction” is a transaction transferring control of an organization, or substantially all assets of one, or subdividing an organization, or merging organizations. If propagation of a covered work results from an entity transaction, each party to that transaction who receives a copy of the work also receives whatever licenses to the work the party’s predecessor in interest had or could give under the previous paragraph, plus a right to possession of the Corresponding Source of the work from the predecessor in interest, if the predecessor has it or can get it with reasonable efforts.

You may not impose any further restrictions on the exercise of the rights granted or affirmed under this License. For example, you may not impose a license fee, royalty, or other charge for exercise of rights granted under this License, and you may not initiate litigation (including a cross-claim or counterclaim in a lawsuit) alleging that any patent claim is infringed by making, using, selling, offering for sale, or importing the Program or any portion of it.

11. Patents.

A “contributor” is a copyright holder who authorizes use under this License of the Program or a work on which the Program is based. The work thus licensed is called the contributor’s “contributor version”.

A contributor’s “essential patent claims” are all patent claims owned or controlled by the contributor, whether already acquired or hereafter acquired, that would be infringed by some manner, permitted by this License, of making, using, or selling its contributor version, but do not include claims that would be infringed only as a consequence of further

modification of the contributor version. For purposes of this definition, “control” includes the right to grant patent sublicenses in a manner consistent with the requirements of this License.

Each contributor grants you a non-exclusive, worldwide, royalty-free patent license under the contributor’s essential patent claims, to make, use, sell, offer for sale, import and otherwise run, modify and propagate the contents of its contributor version.

In the following three paragraphs, a “patent license” is any express agreement or commitment, however denominated, not to enforce a patent (such as an express permission to practice a patent or covenant not to sue for patent infringement). To “grant” such a patent license to a party means to make such an agreement or commitment not to enforce a patent against the party.

If you convey a covered work, knowingly relying on a patent license, and the Corresponding Source of the work is not available for anyone to copy, free of charge and under the terms of this License, through a publicly available network server or other readily accessible means, then you must either (1) cause the Corresponding Source to be so available, or (2) arrange to deprive yourself of the benefit of the patent license for this particular work, or (3) arrange, in a manner consistent with the requirements of this License, to extend the patent license to downstream recipients. “Knowingly relying” means you have actual knowledge that, but for the patent license, your conveying the covered work in a country, or your recipient’s use of the covered work in a country, would infringe one or more identifiable patents in that country that you have reason to believe are valid.

If, pursuant to or in connection with a single transaction or arrangement, you convey, or propagate by procuring

conveyance of, a covered work, and grant a patent license to some of the parties receiving the covered work authorizing them to use, propagate, modify or convey a specific copy of the covered work, then the patent license you grant is automatically extended to all recipients of the covered work and works based on it.

A patent license is “discriminatory” if it does not include within the scope of its coverage, prohibits the exercise of, or is conditioned on the non-exercise of one or more of the rights that are specifically granted under this License. You may not convey a covered work if you are a party to an arrangement with a third party that is in the business of distributing software, under which you make payment to the third party based on the extent of your activity of conveying the work, and under which the third party grants, to any of the parties who would receive the covered work from you, a discriminatory patent license (a) in connection with copies of the covered work conveyed by you (or copies made from those copies), or (b) primarily for and in connection with specific products or compilations that contain the covered work, unless you entered into that arrangement, or that patent license was granted, prior to 28 March 2007.

Nothing in this License shall be construed as excluding or limiting any implied license or other defenses to infringement that may otherwise be available to you under applicable patent law.

12. No Surrender of Others’ Freedom.

If conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot convey a covered work so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a

consequence you may not convey it at all. For example, if you agree to terms that obligate you to collect a royalty for further conveying from those to whom you convey the Program, the only way you could satisfy both those terms and this License would be to refrain entirely from conveying the Program.

13. Use with the GNU Affero General Public License.

Notwithstanding any other provision of this License, you have permission to link or combine any covered work with a work licensed under version 3 of the GNU Affero General Public License into a single combined work, and to convey the resulting work. The terms of this License will continue to apply to the part which is the covered work, but the special requirements of the GNU Affero General Public License, section 13, concerning interaction through a network will apply to the combination as such.

14. Revised Versions of this License.

The Free Software Foundation may publish revised and/or new versions of the GNU General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies that a certain numbered version of the GNU General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that numbered version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of the GNU General Public License, you may choose any version ever published by the Free Software Foundation.

If the Program specifies that a proxy can decide which future versions of the GNU General Public License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Program.

Later license versions may give you additional or different permissions. However, no additional obligations are imposed on any author or copyright holder as a result of your choosing to follow a later version.

15. Disclaimer of Warranty.

THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. Limitation of Liability.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MODIFIES AND/OR CONVEYS THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED

INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

17. Interpretation of Sections 15 and 16.

If the disclaimer of warranty and limitation of liability provided above cannot be given local legal effect according to their terms, reviewing courts shall apply local law that most closely approximates an absolute waiver of all civil liability in connection with the Program, unless a warranty or assumption of liability accompanies a copy of the Program in return for a fee.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively state the exclusion of warranty; and each file should have at least the “copyright” line and a pointer to where the full notice is found.

<one line to give the program’s name and a brief idea of what it does.>

Copyright (C) <year> <name of author>

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program. If not, see <http://www.gnu.org/licenses/>.

Also add information on how to contact you by electronic and paper mail.

If the program does terminal interaction, make it output a short notice like this when it starts in an interactive mode:

<program> Copyright (C) <year> <name of author>

This program comes with ABSOLUTELY NO WARRANTY; for details type `show w'.

This is free software, and you are welcome to redistribute it under certain conditions; type `show c' for details.

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, your program's commands might be different; for a GUI interface, you would use an "about box".

You should also get your employer (if you work as a programmer) or school, if any, to sign a “copyright disclaimer” for the program, if necessary.

For more information on this, and how to apply and follow the GNU GPL, see <http://www.gnu.org/licenses/>.

The GNU General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License. But first, please read <http://www.gnu.org/philosophy/why-not-lgpl.html>.

C. Code listing

Used libraries for the proxy

Non standard Python library:

pyOSC

- http://gitorious.org/pyosc
- License: GNU Lesser General Public License

Used libraries for the control program

controlP5, oscP5, netP5,

- Author: Andreas Schlegel
- http://www.sojamo.de/libraries/controlP5/
- http://www.sojamo.de/libraries/oscP5/
- License: GNU Lesser General Public License

FullScreen API For Processing v 0.98.4

- http://www.superduper.org/processing/fullscreen_api/
- MIT License

D. Code - Proxy

The following pages list the complete python code for the proxy.

PROXY.py

```
1  # Vega - A TUIO proxy with the ability of tangible recognition
2  # Copyright (C) 2012 Thomas Becker
3  # contact: thomas.heinrich.becker@web.de
4
5  # This program is free software: you can redistribute it and/or modify
6  # it under the terms of the GNU General Public License as published by
7  # the Free Software Foundation, either version 3 of the License, or
8  # (at your option) any later version.
9
10 # This program is distributed in the hope that it will be useful,
11 # but WITHOUT ANY WARRANTY; without even the implied warranty of
12 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 # GNU General Public License for more details.
14
15 # You should have received a copy of the GNU General Public License
16 # along with this program.  If not, see <http://www.gnu.org/licenses/>.
17
18 # This is the TUIO proxy main script
19
20 import os
21 from settings           import settings
22 from oscRemote          import remote
23 from tuioTools          import tuioServer
24 from tangiblePattern    import tangibles
25
26
27 # clear screen and print program info
28 os.system('cls' if os.name=='nt' else 'clear')
29
30 print "Vega - TUIO proxy Copyright (C) 2012 Thomas Becker"
31 print "This program comes with ABSOLUTELY NO WARRANTY."
32 print "This is free software, and you are welcome to redistribute it"
33 print "under certain conditions."
34 print ""
35
36 print "TUIO proxy started"
37
38 # load settings from files
39 settings.load()
40
41 # load tangibles from disk
```

PROXY.py

```
42  tangibles.loadTangiblesFromDisk()
43
44  # start the tuio proxy
45  tuioServer.start()
46
47  # start remote control for this programm
48  remote.start()
49
50  print "TUIO proxy up and running..."
51
52  try :
53      while 1 :
54          pass
55
56  except KeyboardInterrupt :
57      tuioServer.stop()
58      remote.stop()
59      print "TUIO proxy aborted by keyboard"
```

tuioTools.py

```
1   # Vega - A TUIO proxy with the ability of tangible
    recognition
2   # Copyright (C) 2012 Thomas Becker
3   # contact: thomas.heinrich.becker@web.de
4
5   # This program is free software: you can redistribute it
    and/or modify
6   # it under the terms of the GNU General Public License as
    published by
7   # the Free Software Foundation, either version 3 of the
    License, or
8   # (at your option) any later version.
9
10  # This program is distributed in the hope that it will be
    useful,
11  # but WITHOUT ANY WARRANTY; without even the implied
    warranty of
12  # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
    See the
13  # GNU General Public License for more details.
14
15  # You should have received a copy of the GNU General
    Public License
16  # along with this program.  If not, see
    <http://www.gnu.org/licenses/>.
17
18  import OSC
19  import threading
20  from settings               import settings
21  from numberGenerator        import numberGenerator
22  from tangiblePattern        import tangibles
23  from trianglePattern        import
    createTrianglesFromCursors
24  from stopwatch              import stopwatch
25  from geometricCalculations  import isPointInCircle
26
27  # a 2D cursor object
28  class two2dCur(object):
29
30      # function called when object is created
31      def __init__(self, *args, **kwargs):
32
33          self.id     = kwargs.get('id', 0)
34          self.x      = kwargs.get('x', 0)
35          self.xRaw   = kwargs.get('xRaw', 0)
36          self.y      = kwargs.get('y', 0)
37          self.vX     = kwargs.get('vX', 0)
38          self.vY     = kwargs.get('vY', 0)
39          self.mA     = kwargs.get('mA', 0)
40
41      # function that makes the object printable
```

tuioTools.py

```
42          def __repr__(self):
43              return "ID: %s x: %s y: %s" % (self.id, self.x,
                self.y)
44
45      # a buffer to store all received TUIO cursors
46      class two2dCurBuffer(object):
47
48          dictOfTuioCursors = {}
49
50          # function to remove to all cursors from buffer if
            they are not in the argument of the function
51          def cleanNonAliveCursors(self,*args):
52              oldCursors = set(self.dictOfTuioCursors.keys())
53              aliveCursors = set(args[0])
54              cursorsToDelete = oldCursors-aliveCursors
55              for c in list(cursorsToDelete):
56                  del self.dictOfTuioCursors[c]
57
58          # function to remove all cursors with IDs provided in
            the argument
59          def removeCursorsWithIDs(self,*args):
60              cursorsToDelete = args[0]
61              if cursorsToDelete != []:
62                  for c in cursorsToDelete:
63                      if c in self.dictOfTuioCursors:
64                          del self.dictOfTuioCursors[c]
65
66          # function to add and change cursors in the buffer
67          def setCursor(self, **kwargs):
68              self.dictOfTuioCursors[kwargs.get('id')] = (
                two2dCur(id=kwargs.get('id'),x=kwargs.get('x'),
69                                                          xRaw=
    kwargs.get('xRaw'),y=kwargs.get('y'),
70                                                          vX=
    kwargs.get('vX'), vY=kwargs.get('vY'),
71                                                          mA=
    kwargs.get('mA')))
72
73          # function to return the IDs of all cursors in the
            buffer
74          def getCursorIDs(self):
75              ids = self.dictOfTuioCursors.keys()
76              return ids
77
78          # function to return the number of alive cursors in
            the buffer
79          def getNumberOfAliveCursors(self):
80              number = len(self.dictOfTuioCursors)
81              return number
82
83          # function returning all objects in the buffer
```

tuioTools.py

```
84          def getCursors(self):
85              cursors = self.dictOfTuioCursors.values()
86              return cursors
87
88      # an object that receives all tuio signals and deals with
        them
89      class tuioServerClass(object):
90
91          # alive point vars
92          alivePointCount = 0
93
94          # the cursor buffer
95          cursorBuffer = two2dCurBuffer()
96
97          # bundles for TUIO messages, one for cursors, one for
            objects
98          twoDcurBundle = OSC.OSCBundle()
99          twoDobjBundle = OSC.OSCBundle()
100
101         # connect the tangibles container function
            getAllCursorsInBuffer with buffer
102         tangibles.registerCursorProvider(cursorBuffer.
            getCursors)
103
104         # function to start the TUIO server
105         def start(self):
106             print "Starting OSC server"
107
108             # setup ips and ports
109             receiveAddress  = settings.get('receiveAddress')
110             receivePort     = settings.get('receivePort')
111             sendAddress     = settings.get('sendAddress')
112             sendPort        = settings.get('sendPort')
113
114             receive_settings    = receiveAddress,   receivePort
115             send_settings       = sendAddress,      sendPort
116
117             # declare server and client
118             self.oscServer = OSC.OSCServer(receive_settings)
119             self.oscClient = OSC.OSCClient()
120             self.oscClient.connect(send_settings)
121
122             # declare message handler
123             self.oscServer.addMsgHandler("/tuio/2Dobj", self.
                twoDobj_handler)
124             self.oscServer.addMsgHandler("/tuio/2Dcur", self.
                twoDcur_handler)
125
126             # start server thread
127             self.oscServerThread = threading.Thread( target =
                self.oscServer.serve_forever )
```

tuioTools.py

```
128             self.oscServerThread.start()
129
130         # function to stop the TUIO server
131         def stop(self):
132             print "Stopping OSC server"
133             self.oscServer.close()
134
135             print "Waiting for server thread to finish"
136             self.oscServerThread.join()
137
138         # function that is called when a message with
            "/tuio/2Dobj" is received
139         def twoDobj_handler(self, addr, tags, payload, source):
140
141             messageType = payload[0]
142
143             if messageType == 'alive':
144                 #print "2Dobj ALIVE", tags, payload[1:]
145                 pass
146
147             elif messageType == 'set':
148                 #print "2Dobj SET"
149                 pass
150
151             elif messageType == 'fseq':
152                 #print "2Dobj FSEQ"
153                 pass
154
155         # function that is called when a message with
            "/tuio/2Dcur" is received
156         def twoDcur_handler(self,addr, tags, payload, source):
157
158             messageType = payload[0]
159
160             if messageType == 'alive':
161                 #start stopwatch
162                 #stopwatch.start()
163
164                 # save number of alive points
165                 self.alivePointCount = tags.count('i')
166                 # delete non alive point from cursor buffer
167                 cursorIDs = payload[1:]
168                 self.cursorBuffer.cleanNonAliveCursors(
                    cursorIDs)
169
170             elif messageType == 'set':
171                 # set 2Dcur in 2Dcur buffer
172                 self.cursorBuffer.setCursor(id = payload[1],
173                                             x = payload[2]*
                                            settings.get(
                                            'touchScreenAspect'
```

tuioTools.py

```
                                        ),
174                                     xRaw = payload[2],
                                        y = payload[3], vX
                                         = payload[4], vY
                                        = payload[5],
175                                     mA = payload[6])
176
177             elif messageType == 'fseq':
178                 #check number of alive points
179                 if self.alivePointCount < 3:
180                     # create 2Dcur bundle with the points in
                        the buffer
181                     self.addAliveIDsTo2DcurBundle()
182                     self.addCursorsTo2DcurBundle()
183                     self.addFseqTo2DcurBundle()
184                     # send 2Dcur bundle
185                     self.send2DcurBundle()
186
187                     # create empty 2Dobj message bundle
188                     self.addEmptyAliveTo2DobjBundle()
189                     self.addFseqTo2DobjBundle()
190                     # send 2Dobj bundle
191                     self.send2DobjBundle()
192
193                     #stopwatch.stop()
194                     #stopwatch.getAverage()
195
196                 else:
197                     cursorsOnScreen = self.cursorBuffer.
                        getCursors()
198                     # compare registered tangible triangles
                        with triangles created at the moment
199                     # calculate tangible postions and rotations
200                     tangibles.checkForKnownTriangles(
                        cursorsOnScreen)
201
202                     # remove points belonging to recognized
                        tangibles from 2Dcur buffer
203                     if (settings.get('tangibleCursorFilter')
                        == 'on'):
204                         self.cursorBuffer.removeCursorsWithIDs
                            (tangibles.getRecognizedCursors())
205
206                     # remove cursors that could be created by
                        fingers touching the tangible
207                     if (settings.get('realFingerFilter') ==
                        'on'):
208                         self.removeRealFingersAroundTangibles()
209
210                     # create 2Dcur bundle with the points in
                        the buffer
```

tuioTools.py

```
211                     self.addAliveIDsTo2DcurBundle()
212                     self.addCursorsTo2DcurBundle()
213                     self.addFseqTo2DcurBundle()
214                     # send 2Dcur bundle
215                     self.send2DcurBundle()
216
217                     # create 2Dobj message bundle with all
                        tangibles
218                     self.addAliveIDsTo2DobjBundle()
219                     self.addObjectsTo2DobjBundle()
220                     self.addFseqTo2DobjBundle()
221                     # send 2Dobj bundle
222                     self.send2DobjBundle()
223
224                     #stopwatch.stop()
225                     #stopwatch.getAverage()
226
227         # function to remove all cursors in a circle around
            the tangible
228         def removeRealFingersAroundTangibles(self):
229             radius = settings.get('realFingerFilterRadius')
230             aspectCorrectionFactor = settings.get(
                'touchScreenAspect')
231
232             cursorsToDelete = []
233
234             for tan in tangibles.getRecognizedTangibles():
235                 #get tangible position
236                 centerX = tan.position.x /
                    aspectCorrectionFactor
237                 centerY = tan.position.y
238
239
240                 # check for all cursors in the 2DCur buffer
                    if they are in a circle around the tangible
                    center
241                 for cursor in self.cursorBuffer.getCursors():
242                     pointX = cursor.x /aspectCorrectionFactor
243                     pointY = cursor.y
244                     if isPointInCircle(centerX, centerY,
                        radius, pointX, pointY,
                        aspectCorrectionFactor):
245                         cursorsToDelete.append(cursor.id)
246
247
248             self.cursorBuffer.removeCursorsWithIDs(
                cursorsToDelete)
249
250         # function to add an initial alive message to the
            cursor bundle
251         def addAliveIDsTo2DcurBundle(self):
```

tuioTools.py

```
252             message = OSC.OSCMessage()
253             message.setAddress("/tuio/2Dcur")
254             message.append('alive')
255             for id in self.cursorBuffer.getCursorIDs():
256                 message.append(id)
257             self.twoDcurBundle.append(message)
258 
259         # function to add all cursors in the buffer to the
            cursor bundle
260         def addCursorsTo2DcurBundle(self):
261             # load the aspect correction to set the TUIO
                cursor x values back to a range between 0 and 1
262             aspectCorrectionFactor = settings.get(
                'touchScreenAspect')
263 
264             for cursor in self.cursorBuffer.getCursors():
265                 message = OSC.OSCMessage()
266                 message.setAddress("/tuio/2Dcur")
267                 message.append('set')
268                 message.append(cursor.id)
269                 message.append(cursor.x /
                    aspectCorrectionFactor)
270                 message.append(cursor.y)
271                 message.append(cursor.vX)
272                 message.append(cursor.vY)
273                 message.append(cursor.mA)
274                 self.twoDcurBundle.append(message)
275 
276         # function to add a seqence message to the cursor
            bundle
277         def addFseqTo2DcurBundle(self):
278             message = OSC.OSCMessage()
279             message.setAddress("/tuio/2Dcur")
280             message.append('fseq')
281             message.append(numberGenerator.newFseq())
282             self.twoDcurBundle.append(message)
283 
284         # function to send the bundle of TUIO cursor messages
            over OSC
285         def send2DcurBundle(self):
286             self.oscClient.send(self.twoDcurBundle)
287             self.twoDcurBundle = OSC.OSCBundle()
288 
289         # function to add an initial alive message to the
            object bundle
290         def addAliveIDsTo2DobjBundle(self):
291             message = OSC.OSCMessage()
292             message.setAddress("/tuio/2Dobj")
293             message.append('alive')
294             for id in tangibles.getRecognizedTangibleIDs():
295                 message.append(id)
```

tuioTools.py

```
296             self.twoDobjBundle.append(message)
297
298         # function to create an empty 2d object bundle header
299         def addEmptyAliveTo2DobjBundle(self):
300             message = OSC.OSCMessage()
301             message.setAddress("/tuio/2Dobj")
302             message.append('alive')
303             self.twoDobjBundle.append(message)
304
305         # function to add all recognized objects in the
            buffer to the object bundle
306         def addObjectsTo2DobjBundle(self):
307             # load the aspect correction to set the TUIO
                cursor x values back to a range between 0 and 1
308             aspectCorrectionFactor = settings.get(
                'touchScreenAspect')
309
310             for tan in tangibles.getRecognizedTangibles():
311                 message = OSC.OSCMessage()
312                 message.setAddress("/tuio/2Dobj")
313                 message.append('set')
314                 message.append(tan.id)
                                                    #s,id
315                 message.append(tan.id)
                                                    #marker Id
316                 message.append(tan.position.x /
                    aspectCorrectionFactor) #x
317                 message.append(tan.position.y)
                                                #y
318                 message.append(tan.rotation)
                                                #angle in radiant
319                 message.append(0.0)
                                                        #vX
320                 message.append(0.0)
                                                        #vY
321                 message.append(0.0)
                                                        #vA
322                 message.append(0.0)
                                                        #m
323                 message.append(0.0)

                    #rotation acceleration
324                 self.twoDobjBundle.append(message)
325
326         # function to add a seqence message to the object
            bundle
327         def addFseqTo2DobjBundle(self):
328             message = OSC.OSCMessage()
329             message.setAddress("/tuio/2Dobj")
330             message.append('fseq')
331             message.append(numberGenerator.newFseq())
```

tuioTools.py

```
332             self.twoDobjBundle.append(message)
333
334         # function to send the bundle of TUIO object messages
            over OSC
335         def send2DobjBundle(self):
336             self.oscClient.send(self.twoDobjBundle)
337             self.twoDobjBundle = OSC.OSCBundle()
338
339         # function to return all cursors curently in the 2D
            cursor buffer
340         def getAllCursorsInBuffer(self):
341             return self.cursorBuffer.values()
342
343     # creates an instance of the tuioServerClass that can be
        imported
344     tuioServer = tuioServerClass()
```

trianglePattern.py

```
1  # Vega - A TUIO proxy with the ability of tangible recognition
2  # Copyright (C) 2012 Thomas Becker
3  # contact: thomas.heinrich.becker@web.de
4
5  # This program is free software: you can redistribute it and/or modify
6  # it under the terms of the GNU General Public License as published by
7  # the Free Software Foundation, either version 3 of the License, or
8  # (at your option) any later version.
9
10 # This program is distributed in the hope that it will be useful,
11 # but WITHOUT ANY WARRANTY; without even the implied warranty of
12 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 # GNU General Public License for more details.
14
15 # You should have received a copy of the GNU General Public License
16 # along with this program.  If not, see <http://www.gnu.org/licenses/>.
17
18 import math
19 import itertools
20 from geometricCalculations import distance
21
22 # class describing a triangle
23 class triangle(object):
24
25     # The side between vertices 1 and 2 has to be the shortest side
26     # The side between vertices 2 and 3 has to be he intermediate side
27     # The side between vertices 3 and 1 has to be the longest side
28
29     counterClockwiseRotation    = True
30
31     # function calculate the side lengths and orientation of the trinagle
32     def setCursors(self, p1, p2, p3):
33
34         # get the length of the triangle sides and put them in order
35         a = distance(p1, p2) + 0.000000001 # prevent same size sides
36         b = distance(p2, p3)
```

trianglePattern.py

```
37          c = distance(p3, p1) - 0.000000001 # prevent same
            size sides
38
39          #
40          if (a > b) & (a > c):
41              # a is longest
42              if b > c:
43                  # b is middle
44                  # c is shortest
45                  self.v1 = p1
46                  self.v2 = p3
47                  self.v3 = p2
48                  self.lengthLongSide     = a
49                  self.lengthMiddleSide   = b
50                  self.lengthShortSide    = c
51              else:
52                  # c is middle
53                  # b is shortest
54                  self.v1 = p2;
55                  self.v2 = p3;
56                  self.v3 = p1;
57                  self.lengthLongSide     = a
58                  self.lengthMiddleSide   = c
59                  self.lengthShortSide    = b
60
61          elif (b > a) & (b > c):
62              # b is longest
63              if a > c:
64                  # a is middle
65                  # c is shortest
66                  self.v1 = p3
67                  self.v2 = p1
68                  self.v3 = p2
69                  self.lengthLongSide     = b
70                  self.lengthMiddleSide   = a
71                  self.lengthShortSide    = c
72              else:
73                  # c is middle
74                  # a is shortest
75                  self.v1 = p2
76                  self.v2 = p1
77                  self.v3 = p3
78                  self.lengthLongSide     = b
79                  self.lengthMiddleSide   = c
80                  self.lengthShortSide    = a
81
82          elif (c > a) & (c > b):
83              # c is longest
84              if a > b:
85                  # a is middle
86                  # b is shortest
```

trianglePattern.py

```
87                      self.v1 = p3
88                      self.v2 = p2
89                      self.v3 = p1
90                      self.lengthLongSide     = c
91                      self.lengthMiddleSide   = a
92                      self.lengthShortSide    = b
93                  else:
94                      # b is middle
95                      # a is shortest
96                      self.v1 = p1
97                      self.v2 = p2
98                      self.v3 = p3
99                      self.lengthLongSide     = c
100                     self.lengthMiddleSide   = b
101                     self.lengthShortSide    = a
102
103             # check for rotation order (v2 left or right of
                v1 to v3)
104             if ((self.v3.x - self.v1.x)*(self.v2.y - self.v1.y
                ) - (self.v3.y - self.v1.y)*(self.v2.x - self.v1.x
                )) > 0:
105                 self.counterClockwiseRotation = True
106             else:
107                 self.counterClockwiseRotation = False
108
109         # function to return the IDs of the cursor points
            building the triangle
110         def getIDs(self):
111             idList = [self.v1.id, self.v2.id, self.v3.id]
112             return idList
113
114         # function to print a fancy info about the triangle
            to console
115         def prettyPrint(self):
116             print
117             print '--------Triangle -------------------'
118             print 'Ids are:', self.getIDs()
119             print self.v1
120             print self.v2
121             print self.v3
122             print
123             print 'Longest side v3-v1:', self.lengthLongSide
124             print 'Middle side  v2-v3:', self.lengthMiddleSide
125             print 'Shortest side    v1-v2:', self.
                lengthShortSide
126             if self.counterClockwiseRotation:
127                 print 'Middle side is counter clockwise
                    oriented'
128             else:
129                 print 'Middle side is clockwise oriented'
130
```

trianglePattern.py

```
131
132  # function to create all possible triangles from a list
     of points
133  def createTrianglesFromCursors(*args):
134
135      listOfCursors = args[0]
136      triangles = []
137
138      # create  tuples with the length of 3 points, in
     sorted order, no repeated elements
139      for combination in list(itertools.combinations(
     listOfCursors,3)):
140          # create triangles from the tuples
141          t = triangle()
142          t.setCursors(combination[0],combination[1],
         combination[2])
143          triangles.append(t)
144
145      return triangles
146
147  # function to compare two triangles on their similarity
148  def compareTriangles (t1, t2, tolerance):
149      # check if both triangles have the same rotation
150      if t1.counterClockwiseRotation == t2.
     counterClockwiseRotation:
151
152          # check if the difference of the long sides is
         within tolerance
153          longSideDifference = math.fabs(t1.lengthLongSide -
          t2.lengthLongSide)
154          if longSideDifference <= tolerance:
155
156              # check if the difference of the middle sides
             is within tolerance
157              middleSideDifference = math.fabs(t1.
             lengthMiddleSide - t2.lengthMiddleSide)
158              if middleSideDifference <= tolerance:
159
160                  # check if the difference of the short
                 sides is within tolerance
161                  shortSideDifference = math.fabs(t1.
                 lengthShortSide - t2.lengthShortSide)
162                  if shortSideDifference <= tolerance:
163
164                      # triangles are considered equal
165                      return True
166      else:
167          # triangles are considered different
168          return False
```

tangiblePattern.py

```
1  # Vega - A TUIO proxy with the ability of tangible recognition
2  # Copyright (C) 2012 Thomas Becker
3  # contact: thomas.heinrich.becker@web.de
4
5  # This program is free software: you can redistribute it and/or modify
6  # it under the terms of the GNU General Public License as published by
7  # the Free Software Foundation, either version 3 of the License, or
8  # (at your option) any later version.
9
10 # This program is distributed in the hope that it will be useful,
11 # but WITHOUT ANY WARRANTY; without even the implied warranty of
12 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 # GNU General Public License for more details.
14
15 # You should have received a copy of the GNU General Public License
16 # along with this program.  If not, see <http://www.gnu.org/licenses/>.
17
18 from trianglePattern        import createTrianglesFromCursors, compareTriangles
19 from settings               import settings
20 from geometricCalculations  import pointVector, clockwiseDifferenceBetweenAngles, calcClockWiseAngle, rotateVector
21 from filter                 import rotationFilter, positionFilter
22 import pickle
23
24 # class for the cursors a tangible consists of
25 class tangibleCursor(object):
26     # declare variables
27     lastKnownPositionX  = 0
28     lastKnownPositionY  = 0
29     lastLiveID          = 0
30     offsetFromCenterX   = 0
31     offsetFromCenterY   = 0
32     votes = 0
33
34     # function to make the object writable to disk
35     def __getstate__(self):
36         return self.__dict__
37
38     # function to make the object loadable from disk
```

tangiblePattern.py

```
39        def __setstate__(self, d):
40            self.__dict__.update(d)
41
42        # function to initalize the tangible point with
          values from the arguments
43        def __init__(self, *args, **kwargs):
44            self.id = kwargs.get('id', 0)
45            self.x = kwargs.get('x', 0)
46            self.y = kwargs.get('y', 0)
47
48        # function calculating the poinst offset from the
          calibration center
49        def calcOffset(self, calibrationCenter):
50            self.offsetFromCenterX = self.x -
              calibrationCenter.x
51            self.offsetFromCenterY = self.y -
              calibrationCenter.y
52
53        # function to increase the votes to a point by one
54        def vote(self):
55            self.votes = self.votes + 1
56
57        # function to return the points votes
58        def getVotes(self):
59            return self.votes
60
61        # function to dump the points votes
62        def deleteVotes(self):
63            self.votes = 0
64
65        # function returning the last live cursor id
          associated with this tangible cursor
66        def getLastLiveID(self):
67            return self.lastLiveID
68
69        # function to print the points votes, position and
          offset from center
70        def prettyPrint(self):
71            print 'ID:', self.id,'Votes:',self.votes ,
              'Position x:',self.x,'y:',self.y, 'Offset from
              center', self.offsetFromCenterX, self.
              offsetFromCenterY
72
73    # class defining a tangible
74    class tangible(object):
75
76        # function called when object is created
77        def __init__(self):
78
79            self.id                 = 0
80            self.position           = pointVector()
```

tangiblePattern.py

```
81              self.rotation           = 0
82              self.outOfJailCard      = 2
83              self.tangibleCursors        = {}
84              self.tangibleTriangles  = []
85              self.currentlyRecognized = False
86              self.lastSuccesfulRunIDs = []
87              self.externalIDtoTangibleCursorIDdict = {}
88              self.externalIDtoTangibleCursorIDdictReverse = {}
89
90              self.tangiblePositionFilter = positionFilter()
91              self.tangibleRotationFilter = rotationFilter()
92
93          # function to set the tangible up with the cursors in
            the argument
94          def registerCursors(self, *args,**kwargs):
95              self.id = kwargs.get('id', 0)
96              cursors = args[0]
97
98              # set the center for tangible creation, if not
                provdided otherwise
99              calibrationCenter = pointVector(settings.get(
                'calibrationCenter')[0],settings.get(
                'calibrationCenter')[1])
100
101             # add points to dictionary
102             for c in cursors:
103                     self.tangibleCursors[c.id] =
                        tangibleCursor(x=c.x,y=c.y)
104                     self.tangibleCursors[c.id].calcOffset(
                        calibrationCenter)
105             # create triangles from points
106             self.tangibleTriangles =
                createTrianglesFromCursors(cursors)
107
108         # function to vote for points with a IDs in argument
109         def voteForIDs(self, arg):
110             for id in arg:
111                 self.tangibleCursors[id].vote()
112
113         # function to delete the votes of every tangible point
114         def dropVotes(self):
115             for id in self.tangibleCursors.keys():
116                 self.tangibleCursors[id].deleteVotes()
117
118         # function to retrieve the tangible cursor id that
            belongs to a live cursor
119         def externalIDtoTangibleCursorID(self, externalID):
120             if externalID in self.
                externalIDtoTangibleCursorIDdict:
121                 tangibleCursorID = self.
                    externalIDtoTangibleCursorIDdict[externalID]
```

tangiblePattern.py

```
122                 return tangibleCursorID
123             else:
124                 return []
125
126         # function to link live cursors with tangible cursors
127         def combineExternalIdWithTangibleCursorID(self,
            externalID, tangibleCursorID):
128
129             # if one entry(exID) already has this entry (curID)
130             if tangibleCursorID in self.
                externalIDtoTangibleCursorIDdictReverse:
131                 # delete this entry
132                 # get the key the normal dictionary has
133                 externalIDtoDelete = self.
                    externalIDtoTangibleCursorIDdictReverse[
                    tangibleCursorID]
134                 # delete this entry
135                 del self.externalIDtoTangibleCursorIDdict[
                    externalIDtoDelete]
136                 # delete also in the inverse dict
137                 del self.
                    externalIDtoTangibleCursorIDdictReverse[
                    tangibleCursorID]
138
139             if externalID in self.
                externalIDtoTangibleCursorIDdict:
140                 tangibleCursorIDtoDelete = self.
                    externalIDtoTangibleCursorIDdict[externalID]
141                 del self.
                    externalIDtoTangibleCursorIDdictReverse[
                    tangibleCursorIDtoDelete]
142                 del self.externalIDtoTangibleCursorIDdict[
                    externalID]
143
144             self.externalIDtoTangibleCursorIDdict[externalID]
                = tangibleCursorID
145             self.externalIDtoTangibleCursorIDdictReverse[
                tangibleCursorID] = externalID
146
147         # function to retrieve the last live id belonging to
            tangible cursor
148         def tangibleCursorIDtoExternalID(self,
            tangibleCursorID):
149             if tangibleCursorID in self.
                externalIDtoTangibleCursorIDdictReverse:
150                 externalID = self.
                    externalIDtoTangibleCursorIDdictReverse[
                    tangibleCursorID]
151                 return externalID
152             else:
153                 return []
```

tangiblePattern.py

```
154
155         # function to set update the last position where a
            set of tangible cursors has been seen
156         def setLastKnownPositions(self, ids, triangle):
157             self.tangibleCursors[ids[0]].lastKnownPositionX =
                triangle.v1.x
158             self.tangibleCursors[ids[0]].lastKnownPositionY =
                triangle.v1.y
159             self.combineExternalIdWithTangibleCursorID(
                triangle.v1.id, ids[0])
160             self.tangibleCursors[ids[0]].lastLiveID         =
                triangle.v1.id
161
162             self.tangibleCursors[ids[1]].lastKnownPositionX =
                triangle.v2.x
163             self.tangibleCursors[ids[1]].lastKnownPositionY =
                triangle.v2.y
164             self.combineExternalIdWithTangibleCursorID(
                triangle.v2.id, ids[1])
165             self.tangibleCursors[ids[1]].lastLiveID         =
                triangle.v2.id
166
167             self.tangibleCursors[ids[2]].lastKnownPositionX =
                triangle.v3.x
168             self.tangibleCursors[ids[2]].lastKnownPositionY =
                triangle.v3.y
169             self.combineExternalIdWithTangibleCursorID(
                triangle.v3.id, ids[2])
170             self.tangibleCursors[ids[2]].lastLiveID         =
                triangle.v3.id
171
172         # function to compare momentary triangles with the
            triangles defining the pattern
173         def compareTangibleTrianglesAndExternalTriangles(self,
             *args):
174             self.externalCursors = args[0]
175             # create all possible triangles from received
                cursors
176             externalTriangles = createTrianglesFromCursors(
                self.externalCursors)
177
178             tolerance = settings.get('tolerance')
179
180             for externalTriangle in externalTriangles:
181                 for internalTriangle in self.tangibleTriangles:
182
183                     if compareTriangles(externalTriangle,
                        internalTriangle,tolerance):
184                         # set last known position of tangible
                            cursors and vote for them
185                         ids = internalTriangle.getIDs()
```

tangiblePattern.py

```
186                     self.setLastKnownPositions(ids,
                        externalTriangle)
187                     self.voteForIDs(ids)
188
189         # sort points by their received votes
190         sortedKeys = []
191         for key, value in sorted(self.tangibleCursors.
            iteritems(), key=lambda (k,v): (v.votes,k),
            reverse=True):
192             sortedKeys.append(key)
193
194         # check if their have been enough votes
195         highestVotes    = self.tangibleCursors[sortedKeys[
            1]].votes
196         neededVotes     = settings.get('neededVotes')
197         if settings.get('debugVotes'):
198             print "Highest",    highestVotes
199             print "Needed",     neededVotes
200
201         if highestVotes >= neededVotes:
202             self.currentlyRecognized = True
203             self.outOfJailCard = 2
204
205             self.lastSuccesfulRunIDs = []
206             for tangibleCursorID in sortedKeys:
207                 self.lastSuccesfulRunIDs.append(self.
                    tangibleCursorIDtoExternalID(
                    tangibleCursorID))
208
209             # calculate tangible postion and rotation
                with best recognized cursors
210             id1 = sortedKeys[0]
211             id2 = sortedKeys[1]
212             self.calculateTangiblePositionAndRotation(id1,
                id2)
213
214         else:
215             #compare current live IDs with last succeful
                recognized IDs
216             liveIDs = []
217             for externalCursor in self.externalCursors:
218                 liveIDs.append(externalCursor.id)
219             cursorIntersection = filter(set(liveIDs).
                __contains__, self.lastSuccesfulRunIDs)
220             #check if at least two cursors are left
221             if len(cursorIntersection) > 1:
222                 self.currentlyRecognized = True
223                 self.outOfJailCard = 2
224
225                 # take first two live ids and calculate
                    with them
```

tangiblePattern.py

```
226                     id1 = cursorIntersection[0]
227                     id2 = cursorIntersection[1]
228                     self.
                        calculateTangiblePositionAndRotationWithLiv
                        eIDs(id1,id2)
229
230                 else:
231                     # to prevent single "glitches" from
                        disturbing the signal the tangible has
                        some out of jail cards
232                     if self.outOfJailCard > 0:
233                         self.outOfJailCard -= 1
234                         self.currentlyRecognized = True
235                     else:
236                         self.currentlyRecognized = False
237                         if settings.get('debugVotes'):
238                             print 'No match for ID:', self.
                                id
239
240             self.dropVotes()
241
242         # function to calculate the current position and
            rotation out of two known tangible cursors
243         def calculateTangiblePositionAndRotationWithLiveIDs(
            self,id1,id2) :
244
245             #translate from live ID to internal ID
246             internalCursorID1 = self.
                externalIDtoTangibleCursorID(id1)
247             internalCursorID2 = self.
                externalIDtoTangibleCursorID(id2)
248
249             # create dictionary with live cursors
250
251             liveCursors = {}
252             for c in self.externalCursors:
253                 liveCursors[c.id] = c
254
255             # calculate original rotation angle
256             p1old = pointVector(self.tangibleCursors[
                internalCursorID1].offsetFromCenterX, self.
                tangibleCursors[internalCursorID1].
                offsetFromCenterY)
257             p2old = pointVector(self.tangibleCursors[
                internalCursorID2].offsetFromCenterX, self.
                tangibleCursors[internalCursorID2].
                offsetFromCenterY)
258             rotationAngleInCenteredTangible =
                calcClockWiseAngle(p1old,p2old)
259
260             # calculate the current angle
```

tangiblePattern.py

```
261
262             p1now = pointVector(liveCursors[id1].x,
                liveCursors[id1].y)
263             p2now = pointVector(liveCursors[id2].x,
                liveCursors[id2].y)
264             rotationAngleOfTangibleNow = calcClockWiseAngle(
                p1now, p2now);
265
266             # calculate the difference between the two angles
267             currentRotation = clockwiseDifferenceBetweenAngles
                (rotationAngleInCenteredTangible,
                rotationAngleOfTangibleNow);
268
269             # check if the rotation filter is set to pre
270             if settings.get('rotationFilterPosition') == 'pre':
271                 # add current rotation value to the rotation
                    filter
272                 self.tangibleRotationFilter.addValue(
                    currentRotation)
273                 # get rotation value from filter
274                 currentRotation = self.tangibleRotationFilter.
                    getState()
275
276             # calculate the vector form current p1 to the
                tangible center
277             shiftOfId1 = rotateVector(p1old, currentRotation)
278             # calculate position
279             currentPosition = p1now - shiftOfId1
280
281             # check if the position filter is active
282             if settings.get('positionFilterActive'):
283                 # add current position to filter
284                 self.tangiblePositionFilter.addXvalue(
                    currentPosition.x)
285                 self.tangiblePositionFilter.addYvalue(
                    currentPosition.y)
286                 # get position from filter
287                 currentPosition.x = self.
                    tangiblePositionFilter.getXstate()
288                 currentPosition.y = self.
                    tangiblePositionFilter.getYstate()
289
290             # check if post rotation filter is active
291             if settings.get('rotationFilterPosition') ==
                'post':
292                 # add current rotation value to the rotation
                    filter
293                 self.tangibleRotationFilter.addValue(
                    currentRotation)
294                 # get rotation value from filter
295                 currentRotation = self.tangibleRotationFilter.
```

```
                getState()
296
297             # set position and rotation
298             self.position = currentPosition
299             self.rotation = currentRotation
300
301         # function to calculate the current position and
            rotation out of two known tangible cursors
302         def calculateTangiblePositionAndRotation(self,id1,id2)
             :
303
304             # calculate original rotation angle
305             p1old = pointVector(self.tangibleCursors[id1].
                offsetFromCenterX,self.tangibleCursors[id1].
                offsetFromCenterY)
306             p2old = pointVector(self.tangibleCursors[id2].
                offsetFromCenterX,self.tangibleCursors[id2].
                offsetFromCenterY)
307             rotationAngleInCenteredTangible =
                calcClockWiseAngle(p1old,p2old)
308
309             # calculate the current angle
310             p1now = pointVector(self.tangibleCursors[id1].
                lastKnownPositionX,self.tangibleCursors[id1].
                lastKnownPositionY)
311             p2now = pointVector(self.tangibleCursors[id2].
                lastKnownPositionX,self.tangibleCursors[id2].
                lastKnownPositionY)
312             rotationAngleOfTangibleNow = calcClockWiseAngle(
                p1now, p2now);
313
314             # calculate the difference between the two angles
315             currentRotation = clockwiseDifferenceBetweenAngles
                (rotationAngleInCenteredTangible,
                rotationAngleOfTangibleNow);
316
317             # check if the rotation filter is set to pre
318             if settings.get('rotationFilterPosition') == 'pre':
319                 # add current rotation value to the rotation
                    filter
320                 self.tangibleRotationFilter.addValue(
                    currentRotation)
321                 # get rotation value from filter
322                 currentRotation = self.tangibleRotationFilter.
                    getState()
323
324             # calculate the vector form current p1 to the
                tangible center
325             shiftOfId1 = rotateVector(p1old, currentRotation)
326             # calculate position
327             currentPosition = p1now - shiftOfId1
```

tangiblePattern.py

```
328
329             # check if the position filter is active
330             if settings.get('positionFilterActive'):
331                 # add current position to filter
332                 self.tangiblePositionFilter.addXvalue(
                    currentPosition.x)
333                 self.tangiblePositionFilter.addYvalue(
                    currentPosition.y)
334                 # get position from filter
335                 currentPosition.x = self.
                    tangiblePositionFilter.getXstate()
336                 currentPosition.y = self.
                    tangiblePositionFilter.getYstate()
337
338             # check if post rotation filter is active
339             if settings.get('rotationFilterPosition') ==
                'post':
340                 # add current rotation value to the rotation
                    filter
341                 self.tangibleRotationFilter.addValue(
                    currentRotation)
342                 # get rotation value from filter
343                 currentRotation = self.tangibleRotationFilter.
                    getState()
344
345             # set position and rotation
346             self.position = currentPosition
347             self.rotation = currentRotation
348
349         # function returning a list of all live ids that have
            been lately identified to belong to the tangible
350         def getLiveCursorIDs(self):
351             cursors = []
352             for c in self.tangibleCursors:
353                 id = self.tangibleCursors[c].getLastLiveID()
354                 cursors.append(id)
355             return cursors
356
357         # function to make the tangible savable to disk
358         def __getstate__(self):
359             try:
360                 list = []
361                 list.append(self.__dict__)
362                 list.append(self.tangibleCursors)
363                 list.append(self.
                    externalIDtoTangibleCursorIDdict)
364                 list.append(self.
                    externalIDtoTangibleCursorIDdictReverse)
365                 return list
366
367             except Exception, e:
```

tangiblePattern.py

```
368             print "Failure: %s" % e
369             print "in function tangible getstate"
370 
371     # function to make the tangbile loadable from disk
372     def __setstate__(self, d):
373         try:
374             self.__dict__.update(d[0])
375             self.tangibleCursors
                                              = d[1]
376             self.externalIDtoTangibleCursorIDdict
                          = d[2]
377             self.
                externalIDtoTangibleCursorIDdictReverse    = d
                [3]
378 
379         except Exception, e:
380             print "Failure: %s" % e
381             print "in function tangible setstate"
382 
383 # class to store the tangible objects
384 class tangibleContainer(object):
385 
386     # function is called when the container is created
387     def __init__(self):
388         self.dictOfTangibles = {}
389 
390     # function to check if tangibles can be recognized in
        the current cursors
391     def checkForKnownTriangles(self,*args):
392         trianglesOnScreen = args[0]
393         for tan in self.dictOfTangibles:
394             self.dictOfTangibles[tan].
                compareTangibleTrianglesAndExternalTriangles(
                trianglesOnScreen)
395 
396     # function to return the IDs of all recognized
        tangibles
397     def getRecognizedTangibleIDs(self):
398         ids = []
399         for tan in self.dictOfTangibles:
400             if (self.dictOfTangibles[tan].
                currentlyRecognized == True):
401                 ids = ids + [tan]
402         return ids
403 
404     # function to return the recognized tangibles
405     def getRecognizedTangibles(self):
406         recognizedTangibles = []
407         for tan in self.dictOfTangibles:
408             if (self.dictOfTangibles[tan].
                currentlyRecognized == True):
```

tangiblePattern.py

```
409                     recognizedTangibles.append(self.
                        dictOfTangibles[tan])
410             return recognizedTangibles
411
412         # function returning all cursors that are part of the
            recognized tangibles
413         def getRecognizedCursors(self):
414             recognizedCursors = []
415             for tan in self.dictOfTangibles:
416                 # get cursor IDs from tangible
417                 recognizedCursors = recognizedCursors + self.
                    dictOfTangibles[tan].getLiveCursorIDs()
418             return recognizedCursors
419
420         # register the getAllCursorsInBuffer function
421         def registerCursorProvider(self, cursorProvider):
422             self.getAllCursorsInBuffer = cursorProvider
423
424         # function to register a new tangible with a certain id
425         def registerNewTangible(self,id):
426             tan = tangible()
427             tan.registerCursors(self.getAllCursorsInBuffer(),
                id=id)
428             self.dictOfTangibles[id] = tan
429             print 'Tangible added, ID:', id
430
431         # function to delete a tangible with a certain id
432         def deleteTangible(self,id):
433             try:
434                 del self.dictOfTangibles[id]
435                 print 'Tangible deleted, ID:',id
436             except:
437                 print "Deletion of tangible failed, no
                    tangible with ID", id
438
439         # function to delete all tangibles (not deleted from
            hard disk!)
440         def deleteAllTangibles(self):
441             self.dictOfTangibles.clear()
442             print 'All tangibles deleted'
443
444         # function to save all tangibles in memory to disk
445         def saveTangiblesToDisk(self):
446             try:
447                 file = open("tangibles.db", "wb") # wb =
                    write mode
448                 pickle.dump(self.dictOfTangibles, file)
449                 file.close()
450                 print "Tangibles written to disk"
451
452             except Exception, e:
```

tangiblePattern.py

```
453                 print "Failure: %s" % e
454                 print 'Could not save tangibles to disk'
455 
456         # function to load tangibles from disk to memory
457         def loadTangiblesFromDisk(self):
458             try:
459                 file = open("tangibles.db", "rb") # read mode
460                 self.dictOfTangibles = pickle.load(file)
461                 print 'Tangibles loaded from disk'
462 
463             except Exception, e:
464                 print "Failure: %s" % e
465                 print 'Loading tangibles from file failed.'
466 
467     # create a container object that can be imported
468     tangibles = tangibleContainer()
```

geometricCalculations.py

```
1  # Vega - A TUIO proxy with the ability of tangible recognition
2  # Copyright (C) 2012 Thomas Becker
3  # contact: thomas.heinrich.becker@web.de
4
5  # This program is free software: you can redistribute it and/or modify
6  # it under the terms of the GNU General Public License as published by
7  # the Free Software Foundation, either version 3 of the License, or
8  # (at your option) any later version.
9
10 # This program is distributed in the hope that it will be useful,
11 # but WITHOUT ANY WARRANTY; without even the implied warranty of
12 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 # GNU General Public License for more details.
14
15 # You should have received a copy of the GNU General Public License
16 # along with this program.  If not, see <http://www.gnu.org/licenses/>.
17
18 import math
19
20 # class of a 2 dimensional vector
21 class pointVector(object):
22
23     # function called when object is created
24     def __init__(self, *args, **kwargs):
25         if len(args) == 0:
26             self.x = kwargs.get('x', 0)
27             self.y = kwargs.get('y', 0)
28         else:
29             self.x = args[0]
30             self.y = args[1]
31
32     # function to add two vectors
33     def __add__(self, other):
34         return pointVector(x=self.x+other.x,y=self.y+other.y)
35
36     # function to subtract vectors
37     def __sub__(self, other):
38         return pointVector(x=self.x-other.x,y=self.y-other.y)
39
40     # make the vector printable
```

geometricCalculations.py

```
41        def __repr__(self):
42            return "x: %s y: %s" % (self.x, self.y)
43
44    # function to calculate the distance between two points
45    def distance(p1, p2):
46        dist =  math.sqrt( ((p2.x - p1.x)**2) + ((p2.y - p1.y
          )**2));
47        return dist
48
49    # function to calculate the clockwise angle between 12
      o'clock (on screen)coming out of p1 and p1-p2
50    def calcClockWiseAngle(p1, p2):
51        origin = p1
52        a = pointVector(p1.x,-500.0)
53        b = p2
54        v1 = a - origin
55        v2 = b - origin
56        angle = -math.atan2(v2.x*v1.y - v2.y*v1.x, v2.x*v1.x+
          v2.y*v1.y)
57
58        if (angle < 0):
59            angle += 2*math.pi
60
61        return angle
62
63    # function to rotate a vector by a certain angle
64    def rotateVector (vec, angle):
65        rotatedVector = pointVector()
66        cosAngle = math.cos(angle)
67        sinAngle = math.sin(angle)
68        rotatedVector.x = vec.x*cosAngle - vec.y*sinAngle
69        rotatedVector.y = vec.x*sinAngle + vec.y*cosAngle
70        return rotatedVector
71
72    # function to calculate the clockwise difference between
      two angles
73    def clockwiseDifferenceBetweenAngles(originalAngle,
      rotatedAngle):
74        if originalAngle > rotatedAngle:
75            rotatedAngle += 2*math.pi
76        angle = rotatedAngle - originalAngle
77        return angle
78    # function to check if a point lies in a circle
79    def isPointInCircle(centerX, centerY, radius, pointX,
      pointY, aspectCorrectionFactor):
80        distanceFromCenter = math.sqrt(((centerX - pointX)*
          aspectCorrectionFactor) ** 2  + (centerY - pointY) **
          2)
81        return distanceFromCenter <= radius
```

settings.py

```
1   # Vega - A TUIO proxy with the ability of tangible recognition
2   # Copyright (C) 2012 Thomas Becker
3   # contact: thomas.heinrich.becker@web.de
4
5   # This program is free software: you can redistribute it and/or modify
6   # it under the terms of the GNU General Public License as published by
7   # the Free Software Foundation, either version 3 of the License, or
8   # (at your option) any later version.
9
10  # This program is distributed in the hope that it will be useful,
11  # but WITHOUT ANY WARRANTY; without even the implied warranty of
12  # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13  # GNU General Public License for more details.
14
15  # You should have received a copy of the GNU General Public License
16  # along with this program.  If not, see <http://www.gnu.org/licenses/>.
17
18  import OSC
19  import pickle
20  from decimal import *
21
22  # class that holds all settings, saves and loads them
23  class settingsManager(object):
24
25      # function called when object is created
26      def __init__(self, filename):
27          self.fileName = filename
28          self.settingsDictionary = {}
29          self.default = {}
30
31      # function to get a certain setting
32      def get(self, arg):
33          return self.settingsDictionary[arg]
34
35      # function to set a certain setting
36      def set(self, name, value):
37          self.settingsDictionary[name] = value
38          print name,'changed to', value
39
40      # function to load settings from disk
41      def load(self):
42          try:
```

settings.py

```
43              file = open(self.fileName, "rb")
44              self.settingsDictionary = pickle.load(file)
45              print self.fileName + ' loaded from disk'
46          except:
47              print 'Loading of ' + self.fileName + '
                failed.'
48              # load default values instead
49              self.setDefaults()
50
51      # function to save settings to disk
52      def save(self):
53          try:
54              file = open(self.fileName, "wb") # write mode
55              pickle.dump(self.settingsDictionary, file)
56              file.close()
57              print "Settings written to " +self.fileName
58          except:
59              print 'Could not write ' + self.fileName + '
                to disk'
60
61      # function to load the default values
62      def setDefaults(self):
63          self.settingsDictionary = self.default
64          print 'Default values loaded'
65
66      # function to return the whole dictionary of settings
67      def returnSettings(self):
68          return self.settingsDictionary
69
70  settings = settingsManager('settings.cfg')
71  # tuio proxy settings
72  settings.default['receiveAddress']                  =
    '127.0.0.1'
73  settings.default['receivePort']                     = 3332
74  settings.default['sendAddress']                     =
    '127.0.0.1'
75  settings.default['sendPort']                        = 3333
76  # remote control settings
77  settings.default['remoteIncomingPort']              = 3330
78  settings.default['remoteControlPort']               = 3331
79  settings.default['remoteControlAddress']            =
    '127.0.0.1'
80  # touch screen settings
81  settings.default['touchScreenAspect']               = (
    16.0/9.0)
82  settings.default['calibrationCenter']               = (0.5
    *(16.0/9.0),0.5)
83  # triangle settings
84  settings.default['tolerance']                       = 0.008
85  settings.default['neededVotes']                     = 1
86  settings.default['debugVotes']                      = 0
```

settings.py

```
87
88  # Position filter settings
89  settings.default['positionFilterActive']          = 0
90  # rotation filter settings
91  settings.default['rotationFilterPosition']        =
    'off' #pre/post/off
92  settings.default['rotationLimit']                 = 0.05
     # limit in rad
93  # filter for tangible fingers settings
94  settings.default['tangibleCursorFilter']          =
    'off' #on/off
95  # filter for real fingers arround tangible
96  settings.default['realFingerFilter']              =
    'off' #on/off
97  settings.default['realFingerFilterRadius']        = 0.3
```

oscRemote.py

```
1  # Vega - A TUIO proxy with the ability of tangible recognition
2  # Copyright (C) 2012 Thomas Becker
3  # contact: thomas.heinrich.becker@web.de
4
5  # This program is free software: you can redistribute it and/or modify
6  # it under the terms of the GNU General Public License as published by
7  # the Free Software Foundation, either version 3 of the License, or
8  # (at your option) any later version.
9
10 # This program is distributed in the hope that it will be useful,
11 # but WITHOUT ANY WARRANTY; without even the implied warranty of
12 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 # GNU General Public License for more details.
14
15 # You should have received a copy of the GNU General Public License
16 # along with this program.  If not, see <http://www.gnu.org/licenses/>.
17
18 import OSC
19 import time, threading
20 from settings           import settings
21 from tangiblePattern    import tangibles
22
23 class remoteControlOverOSC(object):
24
25     # function to start the OSC remote
26     def start(self):
27
28         # setup ips and ports
29         receiveAddress      = '127.0.0.1'
30         receivePort         = settings.get('remoteIncomingPort')
31         receive_settings    = receiveAddress,   receivePort
32
33         # declare server
34         self.oscServer              = OSC.OSCServer(receive_settings)
35
36         # declare message handlers
37         self.oscServer.addMsgHandler("/vega/tangible", self.tangibleRegistration_handler)
38         self.oscServer.addMsgHandler("/vega/ping"
```

oscRemote.py

```
                                    , self.ping_handler)
39          self.oscServer.addMsgHandler("/vega/settings"
                              , self.settings_handler)
40          self.oscServer.addMsgHandler(
            "/vega/requestForSettings"          , self.
            settingsRequest_handler)
41
42          # start server thread
43          self.oscServerThread = threading.Thread( target =
            self.oscServer.serve_forever )
44          self.oscServerThread.start()
45
46      # function to stop the OSC rermote
47      def stop(self):
48          print "Stopping remote OSC server"
49          self.oscServer.close()
50          print "Waiting for remote server thread to finish"
51          self.oscServerThread.join()
52
53      # function that is called when a message with
        settings are send to the proxy, also used to save
        settings to disk
54      def settings_handler(self,addr, tags, stuff, source):
55          if stuff[0] == 'save':
56              settings.save()
57          else:
58              settings.set(stuff[0],stuff[1])
59
60      # function that is called when the remote control
        program request the proxy settings
61      def settingsRequest_handler(self,addr, tags, stuff,
        source):
62          try:
63              settingDict = settings.returnSettings()
64              for key in settingDict:
65                  client = OSC.OSCClient()
66                  msg = OSC.OSCMessage()
67                  msg.setAddress("/vega/settings")
68                  msg.append(key)
69                  msg.append(settingDict[key])
70                  client.sendto(msg, (settings.get(
                    'remoteControlAddress'), settings.get(
                    'remoteControlPort')))
71
72          except Exception, e:
73              print 'Sending settings to control programm
                failed'
74              print "Error:", e
75
76      # function that is called when a tangible
        registration message has been send from the remote
```

oscRemote.py

```
        program
77      def tangibleRegistration_handler(self,addr, tags,
        stuff, source):
78          messageType = stuff[0]
79          if messageType == 'register':
80              print 'Register tangible request received'
81              id = stuff[1]
82              tangibles.registerNewTangible(id)
83
84          elif messageType == 'delete':
85              print 'Delete tangible request received'
86              id = stuff[1]
87              tangibles.deleteTangible(id)
88
89          elif messageType == 'deleteAll':
90              print 'Delete all tangibles request received'
91              tangibles.deleteAllTangibles()
92
93          elif messageType == 'saveToDisk':
94              print 'Save to disk request received'
95              tangibles.saveTangiblesToDisk()
96
97      # function send an alive signal back to the remote
        program
98      def ping_handler(self,addr, tags, stuff, source):
99
100         # save address where ping came from
101         pingSource = OSC.getUrlStr(source).split(':')[0]
102         if settings.get('remoteControlAddress') !=
            pingSource:
103             settings.set('remoteControlAddress',pingSource)
104
105         # read the port of the remote control programm
            from settings
106         port = settings.get('remoteControlPort')
107
108         # send pong message back
109         client = OSC.OSCClient()
110         msg = OSC.OSCMessage()
111         msg.setAddress("/vega/pong")
112         msg.append(1234)
113         client.sendto(msg, (pingSource, port))
114
115 # create the remote control object
116 remote = remoteControlOverOSC()
117
```

numberGenerator.py

```
1  # Vega - A TUIO proxy with the ability of tangible recognition
2  # Copyright (C) 2012 Thomas Becker
3  # contact: thomas.heinrich.becker@web.de
4
5  # This program is free software: you can redistribute it and/or modify
6  # it under the terms of the GNU General Public License as published by
7  # the Free Software Foundation, either version 3 of the License, or
8  # (at your option) any later version.
9
10 # This program is distributed in the hope that it will be useful,
11 # but WITHOUT ANY WARRANTY; without even the implied warranty of
12 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 # GNU General Public License for more details.
14
15 # You should have received a copy of the GNU General Public License
16 # along with this program.  If not, see <http://www.gnu.org/licenses/>.
17
18 class numberGenerator(object):
19
20     #start values
21     touchIdCounter = 1000
22     fseqCounter = 1000
23
24     def newTouchId(self):
25         self.touchIdCounter += 1
26         return self.touchIdCounter
27
28
29     def newFseq(self):
30         self.fseqCounter = self.fseqCounter + 1
31         return self.fseqCounter
32
33 numberGenerator     = numberGenerator()
```

filter.py

```
1  # Vega - A TUIO proxy with the ability of tangible
   recognition
2  # Copyright (C) 2012 Thomas Becker
3  # contact: thomas.heinrich.becker@web.de
4
5  # This program is free software: you can redistribute it
   and/or modify
6  # it under the terms of the GNU General Public License as
   published by
7  # the Free Software Foundation, either version 3 of the
   License, or
8  # (at your option) any later version.
9
10 # This program is distributed in the hope that it will be
   useful,
11 # but WITHOUT ANY WARRANTY; without even the implied
   warranty of
12 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
   See the
13 # GNU General Public License for more details.
14
15 # You should have received a copy of the GNU General
   Public License
16 # along with this program.  If not, see
   <http://www.gnu.org/licenses/>.
17
18 import math
19 from settings           import settings
20
21 # filter for "errors" in the position recognition, not
   implemented anymore (not needed...)
22 class positionFilter(object):
23
24     x = 0.0
25     y = 0.0
26
27     def addXvalue(self, x):
28         self.x = x
29
30     def addYvalue(self, y):
31         self.y = y
32
33     def getXstate(self):
34         return self.x
35
36     def getYstate(self):
37         return self.y
38
39 # a simple low pass rotation filter class
40 class rotationFilter(object):
41
```

filter.py

```
42      # function that is called when object is created
43      def __init__(self):
44          self.filterOutput = 0
45
46      # function to limit the change in rotation
47      def limitDifference(self, difference):
48              rotationLimit = settings.get('rotationLimit')
49
50              # limit rotation
51              if difference > rotationLimit:
52                  difference = rotationLimit
53
54              elif difference < -rotationLimit:
55                  difference = -rotationLimit
56
57              # check if threshold is exceeded
58              elif (-0.02 < difference < 0.02):
59                  difference = 0.0
60
61              return difference
62
63      # function to add a value to the low pass filter
64      def addValue(self,externallyCalculatedRotation):
65
66          change = self.radianDifference(self.filterOutput,
            externallyCalculatedRotation)
67
68          change = self.limitDifference(change)
69          result = self.filterOutput + change
70
71          # check if zero or 2PI are crossed
72          if result < 0:
73              self.filterOutput = result + 2*math.pi
74          elif result > (2*math.pi):
75              self.filterOutput = result - 2*math.pi
76          else:
77              self.filterOutput = result
78
79      # function returning the filter output value
80      def getState(self):
81          return float(self.filterOutput)
82
83      # function returning the difference in radian
84      def radianDifference(self, oldAngle, newAngle):
85          difference = newAngle - oldAngle
86          # check if tangible as been turned to the right
            over zero value
87          if (2*(-math.pi)) < difference < -math.pi:
88              difference += 2*math.pi
89
90          # check if tangible as been turned to the left
```

filter.py

```
            over zero value
91          elif math.pi < difference <= 2*math.pi:
92              difference -= 2*math.pi
93
94          return difference
```

stopwatch.py

```
1  # Vega - A TUIO proxy with the ability of tangible recognition
2  # Copyright (C) 2012 Thomas Becker
3  # contact: thomas.heinrich.becker@web.de
4
5  # This program is free software: you can redistribute it and/or modify
6  # it under the terms of the GNU General Public License as published by
7  # the Free Software Foundation, either version 3 of the License, or
8  # (at your option) any later version.
9
10 # This program is distributed in the hope that it will be useful,
11 # but WITHOUT ANY WARRANTY; without even the implied warranty of
12 # MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
13 # GNU General Public License for more details.
14
15 # You should have received a copy of the GNU General Public License
16 # along with this program.  If not, see <http://www.gnu.org/licenses/>.
17
18 import time
19
20 class stopwatchClass(object):
21     starttime       = 0
22     laptime         = 0
23     stoptime        = 0
24     average         = 0
25     averageSum      = 0
26     averageCount    = 0
27     averageMin      = 100.0
28     averageMax      = 0
29
30     def start(self):
31         self.starttime = time.clock()
32         self.laptime = self.starttime
33
34     def stop(self):
35         self.stoptime = time.clock()
36         self.averageSum = self.averageSum + ((self.stoptime - self.starttime)*1000)
37         self.averageCount = self.averageCount + 1
38
39     def getTime(self):
40         milliseconds = ((self.stoptime - self.starttime)*1000)
```

stopwatch.py

```
41            print "%.5f milliseconds" % milliseconds
42            return milliseconds
43
44        def getLap(self):
45            now = time.clock()
46            milliseconds = ((now - self.laptime)*1000)
47            print "Lap: %.5f milliseconds" % milliseconds
48
49            self.averageSum = self.averageSum + milliseconds
50            self.averageCount = self.averageCount + 1
51            if  (0 < milliseconds < self.averageMin):
52                self.averageMin = milliseconds
53
54            if  milliseconds > self.averageMax:
55                self.averageMax = milliseconds
56
57            self.laptime = now
58            return milliseconds
59
60        def getAverage(self):
61            average = self.averageSum / self.averageCount
62            print "Average:", average, "Min:", self.averageMin
              , "Max:", self.averageMax
63            return average
64
65        def reset(self):
66            self.starttime  = 0
67            self.laptime    = 0
68            self.stoptime   = 0
69
70    stopwatch = stopwatchClass()
```

E. Code - Control software

The following pages list the complete java code for the proxy control software.

vegaControl.pde

```
1  import fullscreen.*;
2  import controlP5.*;
3  import oscP5.*;
4  import netP5.*;
5  import TUIO.*;
6  TuioProcessing tuioClient;
7
8  boolean mainTab             = true;
9  boolean connectToProxyTab   = false;
10 boolean selectDisplayTab    = false;
11 boolean frameCalibrationTab = false;
12 boolean registerTangibleTab = false;
13 boolean fineTuneTab         = false;
14
15 boolean connected           = false;
16
17 boolean showTuioObjects     = false;
18 boolean showTuioCursers     = false;
19
20 final int mainTabId = 1;
21 final int connectToProxyTabId = 2;
22 final int selctDisplayTabId = 3;
23 final int registerTangibleTabId = 4;
24 final int fineTuneTabId = 5;
25
26 // fullscreen object
27 SoftFullScreen fs;
28
29 // graphical units
30 int verticalUnit            = 18;
31 int horizontalUnit          = 40;
32 int darkSquareHeight        = 6;
33
34 int normalWindowHeight      = verticalUnit * 21;
35 int normalWindowWidth       = int(normalWindowHeight *
   1.618);
36
37 OscP5 oscP5;
38 NetAddress myRemoteLocation;
39 ControlP5 controlP5;
40 int myColor = color(0, 0, 0);
41
42
43 vegaControl mainSketch = this;
44
45 void setup() {
46   frameRate(120);
47   println (this.getClass());
48   size(normalWindowWidth, normalWindowHeight);
49   fs = new SoftFullScreen(this, 0);
50
```

vegaControl.pde

```
51      // Set title of the window
52      frame.setTitle("Vega Control");
53
54      // Set up the receiver for this programm
55      oscP5 = new OscP5(this, 3331);
56
57      // Address of the receiving proxy
58      myRemoteLocation = new NetAddress("127.0.0.1", 3330);
59
60      controlP5 = new ControlP5(this);
61      setupButtonsAndFields();
62    }
63
64    void draw() {
65      smooth();
66      background(70);
67      noStroke();
68      connectionToProxy.check();
69
70      fill(30);
71      rect(0, 0, width, verticalUnit * darkSquareHeight);
72
73
74      if (registerTangibleTab)
75      {
76        paintRegisterTangibleTab();
77      }
78
79      if (showTuioObjects  && (!mainTab)&& (!selectDisplayTab))
80      {
81
82
83        try {
84          Vector tuioObjectList = tuioClient.getTuioObjects();
85          for (int i=0;i<tuioObjectList.size();i++) {
86            TuioObject tobj = (TuioObject)tuioObjectList.
              elementAt(i);
87
88            pushMatrix();
89            translate(tobj.getScreenX(width), tobj.getScreenY(
              height));
90            rotate(tobj.getAngle());
91
92            int diameter = int(height *(2.0/5.0));
93            // grey arcs
94            noStroke();
95            fill(30);
96            arc(0, 0, diameter, diameter, radians(51), radians
              (90));
97            arc(0, 0, diameter, diameter, radians(206),
              radians(270));
```

vegaControl.pde

```
98            // white arcs
99            fill(220);
100           arc(0, 0, diameter, diameter, radians(274),
              radians(360));
101           arc(0, 0, diameter, diameter, radians(136),
              radians(180));
102
103           fill(255);
104           textSize(26);
105           text(""+tobj.getSymbolID(), (diameter/2+30), 0);
106           popMatrix();
107         }
108       }
109       catch (Exception e) {
110         System.out.println("problem: "+e.getMessage());
111       }
112     }
113
114     if (showTuioCursers && (!mainTab)&& (!selectDisplayTab))
115     {
116
117       try {
118
119         Vector tuioCursorList = tuioClient.getTuioCursors();
120         for (int i=0;i<tuioCursorList.size();i++) {
121           TuioCursor tcur = (TuioCursor)tuioCursorList.
              elementAt(i);
122           Vector pointList = tcur.getPath();
123
124           if (pointList.size()>0) {
125             noStroke();
126             fill(255, 20, 0);
127             ellipse( tcur.getScreenX(width), tcur.getScreenY
              (height), 40, 40);
128             // Point number text
129             stroke(255, 20, 0);
130             textSize(16);
131             text(""+ tcur.getCursorID(), tcur.getScreenX(
              width)+25, tcur.getScreenY(height)+5);
132           }
133         }
134       }
135       catch (Exception e) {
136         System.out.println("problem: "+e.getMessage());
137       }
138     }
139   }
140
141
142   ////////////// functions called by changing switches
143
```

vegaControl.pde

```
144
145   ///// fine tune tab
146
147   void tangibleCursorFilter(boolean theFlag) {
148     String value;
149     if (theFlag)
150     {
151       value = "on";
152     }
153     else
154     {
155       value = "off";
156     }
157     OscMessage myOscMessage = new OscMessage(
        "/vega/settings");
158     myOscMessage.add("tangibleCursorFilter");
159     myOscMessage.add(value);
160     OscP5.flush(myOscMessage, myRemoteLocation);
161   }
162
163   void realFingerFilter(boolean theFlag) {
164     String value;
165     if (theFlag)
166     {
167       value = "on";
168     }
169     else
170     {
171       value = "off";
172     }
173     OscMessage myOscMessage = new OscMessage(
        "/vega/settings");
174     myOscMessage.add("realFingerFilter");
175     myOscMessage.add(value);
176     OscP5.flush(myOscMessage, myRemoteLocation);
177   }
178
179   void realFingerFilterRadius(float radius) {
180     OscMessage myOscMessage = new OscMessage(
        "/vega/settings");
181     myOscMessage.add("realFingerFilterRadius");
182     myOscMessage.add(radius);
183     OscP5.flush(myOscMessage, myRemoteLocation);
184   }
185
186   // rotation filter position is called with events
187
188   void rotationLimit(float speed) {
189     OscMessage myOscMessage = new OscMessage(
        "/vega/settings");
190     myOscMessage.add("rotationLimit");
```

vegaControl.pde

```
191     myOscMessage.add(speed);
192     OscP5.flush(myOscMessage, myRemoteLocation);
193   }
194
195   void positionFilterActive(boolean theFlag) {
196     int value;
197     if (theFlag)
198     {
199       value = 1;
200     }
201     else
202     {
203       value = 0;
204     }
205     OscMessage myOscMessage = new OscMessage(
        "/vega/settings");
206     myOscMessage.add("positionFilterActive");
207     myOscMessage.add(value);
208     OscP5.flush(myOscMessage, myRemoteLocation);
209   }
210
211   void tolerance(float tolerance) {
212     OscMessage myOscMessage = new OscMessage(
        "/vega/settings");
213     myOscMessage.add("tolerance");
214     myOscMessage.add(tolerance);
215     OscP5.flush(myOscMessage, myRemoteLocation);
216   }
217
218   void neededVotes(int neededVotes) {
219     OscMessage myOscMessage = new OscMessage(
        "/vega/settings");
220     myOscMessage.add("neededVotes");
221     myOscMessage.add(neededVotes);
222     OscP5.flush(myOscMessage, myRemoteLocation);
223   }
224
225   void debugVotes(boolean theFlag) {
226     int value;
227     if (theFlag)
228     {
229       value = 1;
230     }
231     else
232     {
233       value = 0;
234     }
235     OscMessage myOscMessage = new OscMessage(
        "/vega/settings");
236     myOscMessage.add("debugVotes");
237     myOscMessage.add(value);
```

vegaControl.pde

```
238     OscP5.flush(myOscMessage, myRemoteLocation);
239   }
240
241
242   void touchScreenAspect(String aspectString) {
243     float aspect = float(aspectString);
244     OscMessage myOscMessage = new OscMessage(
        "/vega/settings");
245     myOscMessage.add("touchScreenAspect");
246     myOscMessage.add(aspect);
247     OscP5.flush(myOscMessage, myRemoteLocation);
248   }
249
250   //////// tangible registration tab
251
252   public void registerTangible(int theValue) {
253     OscMessage myOscMessage = new OscMessage(
        "/vega/tangible");
254     myOscMessage.add("register");
255     myOscMessage.add(Integer.parseInt(tangibleIdField.
        getText()));
256     OscP5.flush(myOscMessage, myRemoteLocation);
257   }
258
259   public void deleteTangible(int theValue) {
260     println("Sending command to delete tangible with id "+
        tangibleIdField.getText());
261     OscMessage myOscMessage = new OscMessage(
        "/vega/tangible");
262     myOscMessage.add("delete");
263     myOscMessage.add(Integer.parseInt(tangibleIdField.
        getText()));
264     OscP5.flush(myOscMessage, myRemoteLocation);
265   }
266
267   public void deleteAllTangibles(int theValue) {
268     println("Sending command to delete all tangibles");
269     OscMessage myOscMessage = new OscMessage(
        "/vega/tangible");
270     myOscMessage.add("deleteAll");
271     OscP5.flush(myOscMessage, myRemoteLocation);
272   }
273
274   public void saveTangibles(int theValue) {
275     println("Sending command to save tangibles");
276     OscMessage myOscMessage = new OscMessage(
        "/vega/tangible");
277     myOscMessage.add("saveToDisk");
278     OscP5.flush(myOscMessage, myRemoteLocation);
279   }
280
```

vegaControl.pde

```
281  void saveAlgorithmSettings() {
282    println("Saving algorithm settings");
283    OscMessage myOscMessage = new OscMessage(
       "/vega/settings");
284    myOscMessage.add("save");
285    OscP5.flush(myOscMessage, myRemoteLocation);
286  }
287  ////////////// show tuio messages
288
289
290  void tuioObjectsToggle(boolean theFlag) {
291    if (theFlag==true) {
292      println("start showing tuio objects");
293      if (showTuioCursers == false)
294      {
295        tuioClient  = new TuioProcessing(this);
296      }
297      showTuioObjects = true;
298    }
299    else {
300      println("stop showing tuio objects");
301      if (showTuioCursers == false)
302      {
303        tuioClient.dispose();
304      }
305      showTuioObjects = false;
306    }
307  }
308
309  void tuioCursersToggle(boolean theFlag) {
310    if (theFlag==true) {
311      println("start showing tuio cursers");
312      if (showTuioObjects == false)
313      {
314        tuioClient  = new TuioProcessing(this);
315      }
316      showTuioCursers = true;
317    }
318    else {
319      println("stop showing tuio cursers");
320      if (showTuioObjects == false)
321      {
322        tuioClient.dispose();
323      }
324      showTuioCursers = false;
325    }
326  }
327
328
329  ////////////////////////////////////
330
```

vegaControl.pde

```
331  void controlEvent(ControlEvent theControlEvent) {
332    if (theControlEvent.isController()) {
333
334    }
335
336    else if (theControlEvent.isTab())
337    {
338      switch(theControlEvent.tab().id()) {
339
340      case mainTabId: // main tab
341        mainTab               = true;
342        connectToProxyTab = false;
343        selectDisplayTab  = false;
344        registerTangibleTab   = false;
345        fineTuneTab           = false;
346        controlP5.controller("tuioObjectsToggle").setVisible
           (false);
347        controlP5.controller("tuioCursersToggle").setVisible
           (false);
348        fullscreen.off();
349        break;
350
351      case connectToProxyTabId:
352        mainTab               = false;
353        connectToProxyTab     = true;
354        selectDisplayTab      = false;
355        registerTangibleTab   = false;
356        fineTuneTab       = false;
357        controlP5.controller("tuioObjectsToggle").setVisible
           (false);
358        controlP5.controller("tuioCursersToggle").setVisible
           (false);
359        fullscreen.off();
360        break;
361
362      case selctDisplayTabId:
363        mainTab               = false;
364        connectToProxyTab = false;
365        selectDisplayTab  = true;
366        registerTangibleTab   = false;
367        fineTuneTab           = false;
368        controlP5.controller("tuioObjectsToggle").setVisible
           (false);
369        controlP5.controller("tuioCursersToggle").setVisible
           (false);
370        fullscreen.off();
371        break;
372
373
374      case registerTangibleTabId:
375        mainTab               = false;
```

vegaControl.pde

```
376         connectToProxyTab = false;
377         selectDisplayTab  = false;
378         registerTangibleTab   = true;
379         fineTuneTab           = false;
380         controlP5.controller("tuioCursersToggle").setVisible
            (true);
381         controlP5.controller("tuioObjectsToggle").setVisible
            (true);
382         fullscreen.on();
383         break;
384
385       case fineTuneTabId:
386         mainTab             = false;
387         connectToProxyTab   = false;
388         selectDisplayTab    = false;
389         registerTangibleTab = false;
390         fineTuneTab         = true;
391         controlP5.controller("tuioCursersToggle").setVisible
            (true);
392         controlP5.controller("tuioObjectsToggle").setVisible
            (true);
393         fullscreen.on();
394         break;
395       }
396
397       println("Switched to tab : " + theControlEvent.tab().
          name());
398     }
399
400     else if (theControlEvent.isGroup()) {
401       if (theControlEvent.group().name() == "displayList") {
402         int dn =  int(theControlEvent.group().value());
403         fullscreen.setScreenNumber(dn);
404         if (fullscreen.isOn() == true)
405         {
406           fullscreen.off();
407           fullscreen.forceOn();
408         }
409       }
410       if (theControlEvent.group().name() ==
          "rotationFilterPosition") {
411         int flag =  int(theControlEvent.group().value());
412         String value = "";
413         switch(flag) {
414         case 0:
415           value = "off";
416           break;
417         case 1:
418           value = "post";
419           break;
420         case 2:
```

vegaControl.pde

```
421             value = "pre";
422             break;
423           }
424
425           OscMessage myOscMessage = new OscMessage(
              "/vega/settings");
426           myOscMessage.add("rotationFilterPosition");
427           myOscMessage.add(value);
428           OscP5.flush(myOscMessage, myRemoteLocation);
429         }
430       }
431     }
432
433
```

fullScreen.pde

```
1  import java.awt.GraphicsEnvironment;
2  import java.awt.GraphicsDevice;
3
4  fullScreenHandler fullscreen = new fullScreenHandler();
5
6  class fullScreenHandler{
7
8    int fullScreenDisplayNumber = 0;
9
10   void on(){
11     if (fs.isFullScreen() == false){
12       fs = new SoftFullScreen(mainSketch,
         fullScreenDisplayNumber);
13       GraphicsEnvironment env = GraphicsEnvironment.
         getLocalGraphicsEnvironment();
14       GraphicsDevice[] devices = env.getScreenDevices();
15       int w = devices[fullScreenDisplayNumber].
         getDisplayMode().getWidth();
16       int h = devices[fullScreenDisplayNumber].
         getDisplayMode().getHeight();
17       frame.setSize(w, h);
18       size (w, h);
19       fs.enter();
20     }
21   }
22
23   void forceOn(){
24       fs = new SoftFullScreen(mainSketch,
         fullScreenDisplayNumber);
25       GraphicsEnvironment env = GraphicsEnvironment.
         getLocalGraphicsEnvironment();
26       GraphicsDevice[] devices = env.getScreenDevices();
27       int w = devices[fullScreenDisplayNumber].
         getDisplayMode().getWidth();
28       int h = devices[fullScreenDisplayNumber].
         getDisplayMode().getHeight();
29       frame.setSize(w, h);
30       size (w, h);
31       fs.enter();
32   }
33
34   boolean isOn(){
35     return fs.isFullScreen();
36   }
37
38   void off(){
39     fs.leave();
40     frame.setSize(normalWindowWidth, normalWindowHeight);
41     size (normalWindowWidth, normalWindowHeight);
42   }
43
```

fullScreen.pde

```
44    void setScreenNumber(int num){
45      this.fullScreenDisplayNumber = num;
46    }
47
48    int numberOfScreens(){
49    GraphicsEnvironment env = GraphicsEnvironment.
      getLocalGraphicsEnvironment();
50    GraphicsDevice[] devices = env.getScreenDevices();
51    println(devices[0].getDisplayMode().getHeight());
52    return devices.length;
53    }
54
55  }
56
```

connection.pde

```
1  int ip0Value;
2  int ip1Value;
3  int ip2Value;
4  int ip3Value;
5
6  connectionChecker connectionToProxy = new
   connectionChecker();
7
8  class connectionChecker
9  {
10   boolean firstConnect = true;
11   boolean connectionAlive = false;
12   int timeout = 100;      // in frames
13   int timeForPing = 10;  // in frames
14   int timer   = 0;
15
16   void check()
17   {
18     timer++;
19
20     if (timer == (timeout - timeForPing))
21     {
22       connectionAlive = false;
23       pingProxy();
24     }
25
26     if (timer > timeout)
27     {
28       connectedToggle.setState(connectionAlive);
29       timer = 0;
30     }
31   }
32
33   void isAlive(boolean value)
34   {
35     if (firstConnect) {
36       getSettingsFromProxy();
37       firstConnect = false;
38     }
39
40     connectionAlive = value;
41   }
42 }
43
44 void pingProxy() {
45   println("Pinging proxy ");
46   OscMessage myOscMessage = new OscMessage("/vega/ping");
47   myOscMessage.add(0);
48   OscP5.flush(myOscMessage, myRemoteLocation);
49 }
50
```

connection.pde

```
51  void ip0 (int ipBlock) {
52    String newIp = ( int(ip0.value()) + "." + int(ip1.value
      ()) + "." + int(ip2.value()) + "." + int(ip3.value()));
53    myRemoteLocation = new NetAddress(newIp, 3330);
54    println("proxy ip changed to " + newIp);
55  }
56
57  void ip1 (int ipBlock) {
58    String newIp = ( int(ip0.value()) + "." + int(ip1.value
      ()) + "." + int(ip2.value()) + "." + int(ip3.value()));
59    myRemoteLocation = new NetAddress(newIp, 3330);
60    println("proxy ip changed to " + newIp);
61  }
62
63  void ip2 (int ipBlock) {
64    String newIp = ( int(ip0.value()) + "." + int(ip1.value
      ()) + "." + int(ip2.value()) + "." + int(ip3.value()));
65    myRemoteLocation = new NetAddress(newIp, 3330);
66    println("proxy ip changed to " + newIp);
67  }
68
69  void ip3 (int ipBlock) {
70    String newIp = ( int(ip0.value()) + "." + int(ip1.value
      ()) + "." + int(ip2.value()) + "." + int(ip3.value()));
71    myRemoteLocation = new NetAddress(newIp, 3330);
72    println("proxy ip changed to " + newIp);
73  }
74
75  void oscEvent(OscMessage theOscMessage) {
76
77    if (theOscMessage.checkAddrPattern("/vega/pong")==true) {
78      connectionToProxy.isAlive(true);
79      println("CONNECTED");
80    }
81
82    if (theOscMessage.checkAddrPattern("/vega/settings")==
      true) {
83      String settingName = theOscMessage.get(0).stringValue
        ();
84
85      if (settingName.equals("tolerance") == true)
86      {
87        float receivedValue = theOscMessage.get(1).
          floatValue();
88        controlP5.controller("tolerance").changeValue(
          receivedValue);
89      }
90
91      if (settingName.equals("neededVotes") == true)
92      {
93        int receivedValue = theOscMessage.get(1).intValue();
```

connection.pde

```
94          controlP5.controller("neededVotes").changeValue(
            receivedValue);
95        }
96
97        if (settingName.equals("debugVotes") == true)
98        {
99          boolean receivedValue = (theOscMessage.get(1).
            intValue() != 0);
100         dv.setValue(receivedValue);
101       }
102
103       if (settingName.equals("rotationFilterPosition") ==
          true)
104       {
105         String receivedValue = (theOscMessage.get(1).
            stringValue());
106         if (receivedValue.equals("off"))
107         {
108           rotationFilterSwitchDrop.setValue(0);
109         }
110         else if (receivedValue.equals("post"))
111         {
112           rotationFilterSwitchDrop.setValue(1);
113         }
114         if (receivedValue.equals("pre"))
115         {
116           rotationFilterSwitchDrop.setValue(2);
117         }
118       }
119
120       if (settingName.equals("rotationLimit") == true)
121       {
122         float receivedValue = theOscMessage.get(1).
            floatValue();
123         controlP5.controller("rotationLimit").changeValue(
            receivedValue);
124       }
125
126       if (settingName.equals("realFingerFilter") == true)
127       {
128         String receivedValue = (theOscMessage.get(1).
            stringValue());
129         if (receivedValue.equals("off"))
130         {
131           rff.setValue(false);
132         }
133         else if (receivedValue.equals("on"))
134         {
135           rff.setValue(true);
136         }
137       }
```

connection.pde

```
138
139       if (settingName.equals("realFingerFilterRadius") ==
          true)
140       {
141         float receivedValue = theOscMessage.get(1).
          floatValue();
142         controlP5.controller("realFingerFilterRadius").
          changeValue(receivedValue);
143       }
144
145       if (settingName.equals("tangibleCursorFilter") == true)
146       {
147         String receivedValue = (theOscMessage.get(1).
          stringValue());
148         if (receivedValue.equals("off"))
149         {
150           tcf.setValue(false);
151         }
152         else if (receivedValue.equals("on"))
153         {
154           tcf.setValue(true);
155         }
156
157       }
158
159       if (settingName.equals("positionFilterActive") == true)
160       {
161         boolean receivedValue = (theOscMessage.get(1).
          intValue() != 0);
162         pfa.setValue(receivedValue);
163       }
164
165       if (settingName.equals("touchScreenAspect") == true)
166       {
167         float receivedValue = (theOscMessage.get(1).
          floatValue());
168         println(receivedValue);
169         aspect.setText(Float.toString(receivedValue));
170       }
171
172     }
173   }
174
175
176   void getSettingsFromProxy()
177   {
178     println("Requesting settings from proxy");
179     OscMessage myOscMessage = new OscMessage(
      "/vega/requestForSettings");
180     myOscMessage.add(0);
181     OscP5.flush(myOscMessage, myRemoteLocation);
```

connection.pde

```
182   }
183
184   void refresh(TuioTime bundleTime) {
185     redraw();
186   }
187
188
```

buttonsAndFields.pde

```
1   Toggle connectedToggle;
2
3   void setupButtonsAndFields() {
4
5       setupMainTab();
6       selectConnectToProxyTab();
7       setupSelectDisplayTab();
8       setupRegisterTangibleTab();
9       setupFineTuneTab();
10
11      //connection indicator
12      connectedToggle = controlP5.addToggle("connected",
        true, 0, verticalUnit, horizontalUnit, verticalUnit);
13      connectedToggle.lock();
14      connectedToggle.setColorActive(color(50, 118, 32));
15      connectedToggle.setColorBackground(color(90, 0, 0));
16      connectedToggle.setCaptionLabel("");
17      connectedToggle.setState(false);
18      connectedToggle.moveTo("global");
19
20      controlP5.addToggle("tuioCursersToggle",false,
        horizontalUnit,(darkSquareHeight + 1) * verticalUnit,
        20,20);
21      controlP5.controller("tuioCursersToggle").setVisible(
        false);
22      controlP5.controller("tuioCursersToggle").moveTo(
        "global");
23      controlP5.controller("tuioCursersToggle").
        setCaptionLabel("Show TUIO cursers");
24
25      controlP5.addToggle("tuioObjectsToggle",false,
        horizontalUnit*4,(darkSquareHeight + 1) * verticalUnit
        ,20,20);
26      controlP5.controller("tuioObjectsToggle").setVisible(
        false);
27      controlP5.controller("tuioObjectsToggle").moveTo(
        "global");
28      controlP5.controller("tuioObjectsToggle").
        setCaptionLabel("Show TUIO objects");
29  }
30
31
```

mainTab.pde

```
1   Textarea mainInfoTextArea;
2
3   Numberbox ip0;
4   Numberbox ip1;
5   Numberbox ip2;
6   Numberbox ip3;
7
8   Textarea ipNoteTextArea;
9
10  Textarea selectDisplayTextArea;
11  DropdownList selectDisplay;
12
13  void setupMainTab()
14  {
15      controlP5.tab("default").setLabel("Main");
16      controlP5.tab("default").activateEvent(true);
17      controlP5.tab("default").setHeight(verticalUnit);
18      controlP5.tab("default").setWidth(horizontalUnit-8
        );
19      controlP5.tab("default").setId(mainTabId);
20
21      // main info text
22      mainInfoTextArea = controlP5.addTextarea(
        "mainInfoText", "Welcome to vegaControl!\n\nJust
        click the tabs on top of this window in their given
        order to setup the proxy", horizontalUnit, 3 *
        verticalUnit, width- 2*horizontalUnit, 200);
23      mainInfoTextArea.moveTo("default");
24  }
```

connectToProxyTab.pde

```
1   Textarea selectIpTextArea;
2   Textarea connectInfoTextArea;
3
4   void selectConnectToProxyTab()
5   {
6       controlP5.tab("connectTab").setLabel("1 - Connect to
        proxy");
7       controlP5.tab("connectTab").setHeight(verticalUnit);
8       controlP5.tab("connectTab").activateEvent(true);
9       controlP5.tab("connectTab").setId(connectToProxyTabId);
10
11      connectInfoTextArea = controlP5.addTextarea(
        "connectInfoText", "Connect to the proxy by selecting
        the IP address\n\nOnce connected the red label
        underneath the \"MAIN\" tab turns green.",
        horizontalUnit, 3 * verticalUnit, width- 2*
        horizontalUnit, 200);
12      connectInfoTextArea.moveTo("connectTab");
13
14      selectIpTextArea = controlP5.addTextarea("ipText",
        "Select the IP of the proxy:", horizontalUnit, (
        darkSquareHeight + 1) * verticalUnit, 200, 200);
15      selectIpTextArea.moveTo("connectTab");
16
17      ip0 = controlP5.addNumberbox("ip0",100,horizontalUnit
        ,(darkSquareHeight + 2) * verticalUnit,horizontalUnit,
        verticalUnit);
18      ip0.setValue(127);
19      ip0.setMin(0);
20      ip0.setMax(255);
21      ip0.captionLabel().setVisible(false);
22      controlP5.controller("ip0").moveTo("connectTab");
23
24      ip1 = controlP5.addNumberbox("ip1",100,2 *
        horizontalUnit + 8,(darkSquareHeight + 2) *
        verticalUnit,horizontalUnit,verticalUnit);
25      ip1.setValue(0);
26      ip1.setMin(0);
27      ip1.setMax(255);
28      ip1.captionLabel().setVisible(false);
29      controlP5.controller("ip1").moveTo("connectTab");
30
31      ip2 = controlP5.addNumberbox("ip2",100,3 *
        horizontalUnit + 16,(darkSquareHeight + 2) *
        verticalUnit,horizontalUnit,verticalUnit);
32      ip2.setValue(0);
33      ip2.setMin(0);
34      ip2.setMax(255);
35      ip2.captionLabel().setVisible(false);
36      controlP5.controller("ip2").moveTo("connectTab");
37
```

connectToProxyTab.pde

```
38        ip3 = controlP5.addNumberbox("ip3",100,4 *
          horizontalUnit + 24,(darkSquareHeight + 2) *
          verticalUnit,horizontalUnit,verticalUnit);
39        ip3.setValue(1);
40        ip3.setMin(0);
41        ip3.setMax(255);
42        ip3.captionLabel().setVisible(false);
43        controlP5.controller("ip3").moveTo("connectTab");
44
45        ipNoteTextArea = controlP5.addTextarea("ipNote",
          "NOTE: Normally you leave this unchanged to 127.0.0.1
          because your proxy is running on the same machine.", 7
           * horizontalUnit, (darkSquareHeight + 2) *
          verticalUnit, 270, 200);
46        ipNoteTextArea.moveTo("connectTab");
47    }
```

selectDisplayTab.pde

```
Textarea selectDisplayInfoTextArea;

void setupSelectDisplayTab()
{
    // connect to proxy tab
    controlP5.tab("displayTab").setLabel("2 - Select
    display");
    controlP5.tab("displayTab").setHeight(verticalUnit);
    controlP5.tab("displayTab").activateEvent(true);
    controlP5.tab("displayTab").setId(selctDisplayTabId);

    // select display info text
    selectDisplayInfoTextArea = controlP5.addTextarea(
    "selectDisplayInfoText", "Select the display where
    the touch-frame is attached to.\n\nThe folowing steps
    will be displayed on this screen", horizontalUnit, 3 *
     verticalUnit, width- 2*horizontalUnit, 200);
    selectDisplayInfoTextArea.moveTo("displayTab");

    // fullscreen display selector
    selectDisplayTextArea = controlP5.addTextarea(
    "calibrationinst", "Select the display with
    touchscreen", horizontalUnit , (darkSquareHeight + 1)
    * verticalUnit, 200, 200);
    selectDisplayTextArea.moveTo("displayTab");

    selectDisplay = controlP5.addDropdownList(
    "displayList", horizontalUnit, (darkSquareHeight + 3)
    * verticalUnit, 160, 100);
    selectDisplay.setBackgroundColor(color(190));
    selectDisplay.setItemHeight(20);
    selectDisplay.setBarHeight(15);
    selectDisplay.captionLabel().set("Display with
    touchscreen");
    selectDisplay.captionLabel().style().marginTop = 3;
    selectDisplay.captionLabel().style().marginLeft = 3;
    selectDisplay.valueLabel().style().marginTop = 3;
    for (int i=0;i<fullscreen.numberOfScreens();i++)
    {
    selectDisplay.addItem("Display "+i, i);
    }
    selectDisplay.setColorBackground(color(60));
    selectDisplay.setColorActive(color(255, 128));
    selectDisplay.moveTo("displayTab");

}
```

registerTangibleTab.pde

```
1  registerCenterStar centerStar = new registerCenterStar();
2  Textfield tangibleIdField;
3  Textarea registerTangibleInfoTextArea;
4
5  void setupRegisterTangibleTab()
6  {
7      // Register tangible tab
8      controlP5.tab("tangible").setLabel("3 - Register
       tangible");
9      controlP5.tab("tangible").activateEvent(true);
10     controlP5.tab("tangible").setHeight(verticalUnit);
11     controlP5.tab("tangible").setId(registerTangibleTabId);
12
13     // Register tangible info text
14     registerTangibleInfoTextArea = controlP5.addTextarea(
       "registerTangibleInfoText", "Enable \"SHOW TUIO
       OBJECTS\" and place the tangible you want to register
       on the center of the pattern. On the left select the
       TUIO object ID and click \"Register tangible\".\n\nTo
       delete a tangible enter its ID and click \"Delete
       tangible\". Once you are satisfied, save them to disk
       for the proxy to know them after a restart.",
       horizontalUnit, 3 * verticalUnit, width- 2*
       horizontalUnit+600, 200);
15     registerTangibleInfoTextArea.moveTo("tangible");
16
17     controlP5.addTextlabel("registerLabel", "Register
       tangibles", horizontalUnit, (darkSquareHeight + 5) *
       verticalUnit);
18     controlP5.controller("registerLabel").moveTo(
       "tangible");
19
20     tangibleIdField = controlP5.addTextfield("tangibleId",
        horizontalUnit, (darkSquareHeight + 6) * verticalUnit
       , 40, 20);
21     tangibleIdField.setCaptionLabel("tangibleId");
22     tangibleIdField.captionLabel().toUpperCase(false);
23     tangibleIdField.setText("44");
24     tangibleIdField.setWidth(horizontalUnit);
25     controlP5.controller("tangibleId").moveTo("tangible");
26
27     controlP5.Button b = controlP5.addButton(
       "registerTangible", 0, horizontalUnit*2 + 8, (
       darkSquareHeight + 6) * verticalUnit, 100, 19);
28     b.setCaptionLabel("Register tangible");
29     b.captionLabel().toUpperCase(false);
30     controlP5.controller("registerTangible").moveTo(
       "tangible");
31
32     controlP5.Button deleteTangible = controlP5.addButton(
       "deleteTangible", 0, horizontalUnit*2 +8, (
```

registerTangibleTab.pde

```
        darkSquareHeight + 8) * verticalUnit, 100, 19);
33      deleteTangible.setCaptionLabel("Delete tangible");
34      deleteTangible.captionLabel().toUpperCase(false);
35      controlP5.controller("deleteTangible").moveTo(
        "tangible");
36
37      controlP5.Button deleteTangibles = controlP5.addButton
        ("deleteAllTangibles", 0, horizontalUnit, (
        darkSquareHeight + 10) * verticalUnit,  100 + 8 +
        horizontalUnit, 19);
38      deleteTangibles.setCaptionLabel("        Delete all
        tangibles");
39      deleteTangibles.captionLabel().toUpperCase(false);
40      controlP5.controller("deleteAllTangibles").moveTo(
        "tangible");
41
42      // save tangibles button
43      controlP5.Button saveTangibles = controlP5.addButton(
        "saveTangibles", 0,  horizontalUnit, (darkSquareHeight
         + 12) * verticalUnit, 100 + 8 +horizontalUnit, 19);
44      saveTangibles.setCaptionLabel("         SAVE TO DISK");
45      saveTangibles.captionLabel().toUpperCase(false);
46      saveTangibles.setColorBackground(color(90, 0, 0));
47      controlP5.controller("saveTangibles").moveTo(
        "tangible");
48  }
49
50  void paintRegisterTangibleTab()
51  {
52      centerStar.paint();
53  }
54
55
56  class registerCenterStar
57  {
58    void paint()
59    {
60      int diameter = int(height *(3.0/5.0));
61      // grey arcs
62      fill(30);
63      arc(width/2, height/2, diameter, diameter, radians(0),
         radians(50));
64      arc(width/2, height/2, diameter, diameter, radians(180
        ), radians(205));
65      // white arcs
66      fill(220);
67      arc(width/2, height/2, diameter, diameter, radians(270
        ), radians(273));
68      arc(width/2, height/2, diameter, diameter, radians(90
        ), radians(135));
69    }
```

registerTangibleTab.pde

```
70    }
71
```

fineTuneTab.pde

```
1   Textarea fineTuneInfoTextArea;
2
3   DropdownList rotationFilterSwitchDrop;
4   Textfield aspect;
5   Toggle dv,rff,tcf, pfa;
6
7   void setupFineTuneTab()
8   {
9     controlP5.tab("fineTune").setLabel("4 - Fine-tune
      recognition");
10    controlP5.tab("fineTune").activateEvent(true);
11    controlP5.tab("fineTune").setHeight(verticalUnit);
12    controlP5.tab("fineTune").setId(fineTuneTabId);
13
14    // fine tune info text
15    fineTuneInfoTextArea = controlP5.addTextarea(
      "fineTuneInfoText", "First tune the tolerance of the
      algorithm. After that block recognitions with not
      enough votes.\n\nEnable and tune position and rotation
      filtering as you wish.", horizontalUnit, 3 *
      verticalUnit, width- 2*horizontalUnit+600, 200);
16    fineTuneInfoTextArea.moveTo("fineTune");
17
18
19
20    // tangible cursor filter
21    controlP5.addTextlabel("CursorFilterInfoText",
      "Tangible cursor filter - removes all cursors belonging
      to detected tangibles (in proxy! don't confuse this
      with SHOW TUIO CURSORS option above which just doesn't
      display the cursors in the config program)",
      horizontalUnit, (darkSquareHeight + 5) * verticalUnit);
22    controlP5.controller("CursorFilterInfoText").moveTo(
      "fineTune");
23
24    tcf = controlP5.addToggle("tangibleCursorFilter", false,
       horizontalUnit, (darkSquareHeight + 6) * verticalUnit,
      20, 20);
25    controlP5.controller("tangibleCursorFilter").setVisible(
      true);
26    controlP5.controller("tangibleCursorFilter").moveTo(
      "fineTune");
27    controlP5.controller("tangibleCursorFilter").
      setCaptionLabel("Filter cursors belonging to tangibles");
28
29    // real finger filter
30    controlP5.addTextlabel("realFingerFilterInfoText",
      "Real finger filter - removes all cursors in a radius
      around the tangible center - good for removing fingers
      grabbing the tangible", horizontalUnit, (
      darkSquareHeight + 12) * verticalUnit);
```

fineTuneTab.pde

```
31      controlP5.controller("realFingerFilterInfoText").moveTo(
        "fineTune");
32
33      rff = controlP5.addToggle("realFingerFilter", false,
        horizontalUnit, (darkSquareHeight + 13) * verticalUnit,
        20, 20);
34      controlP5.controller("realFingerFilter").setVisible(true
        );
35      controlP5.controller("realFingerFilter").moveTo(
        "fineTune");
36      controlP5.controller("realFingerFilter").setCaptionLabel
        ("Filter cursors close to the tangible");
37
38      // real finger filter radius
39      controlP5.addSlider("realFingerFilterRadius", 0.0F, 1.0F
        , 0.3F, horizontalUnit, (darkSquareHeight + 15) *
        verticalUnit, 400, 20);
40      controlP5.controller("realFingerFilterRadius").
        setCaptionLabel("Finger filter radius");
41      controlP5.controller("realFingerFilterRadius").
        captionLabel().toUpperCase(false);
42      controlP5.controller("realFingerFilterRadius").moveTo(
        "fineTune");
43
44      // rotation filter switch
45
46      controlP5.addTextlabel("rotationFilterSwitchInfoText",
        "Rotation filter - A simple low pass filter that limits
        the movement per refresh cycle ", horizontalUnit, (
        darkSquareHeight + 20) * verticalUnit);
47      controlP5.controller("rotationFilterSwitchInfoText").
        moveTo("fineTune");
48
49      rotationFilterSwitchDrop = controlP5.addDropdownList(
        "rotationFilterPosition", horizontalUnit, (
        darkSquareHeight + 23) * verticalUnit, 160, 100);
50      rotationFilterSwitchDrop.setBackgroundColor(color(190));
51      rotationFilterSwitchDrop.setItemHeight(20);
52      rotationFilterSwitchDrop.setBarHeight(20);
53      rotationFilterSwitchDrop.captionLabel().set("Rotation
        filter setting");
54      rotationFilterSwitchDrop.captionLabel().style().
        marginTop = 3;
55      rotationFilterSwitchDrop.captionLabel().style().
        marginLeft = 3;
56      rotationFilterSwitchDrop.valueLabel().style().marginTop
        = 3;
57      rotationFilterSwitchDrop.addItem("OFF", 0);
58      rotationFilterSwitchDrop.addItem("POST", 1);
59      rotationFilterSwitchDrop.addItem("PRE", 2);
60      rotationFilterSwitchDrop.setColorBackground(color(60));
```

fineTuneTab.pde

```
61      rotationFilterSwitchDrop.setColorActive(color(255, 128));
62      rotationFilterSwitchDrop.moveTo("fineTune");
63
64      // rotation filter limiter
65      controlP5.addSlider("rotationLimit", 0.0F, 0.2F, 0.05F,
        horizontalUnit, (darkSquareHeight + 27) * verticalUnit,
        400, 20);
66      controlP5.controller("rotationLimit").setCaptionLabel(
        "Rotation filter limiter");
67      controlP5.controller("rotationLimit").captionLabel().
        toUpperCase(false);
68      controlP5.controller("rotationLimit").moveTo("fineTune");
69
70      // position filter active
71      controlP5.addTextlabel("positionFilterInfoText",
        "Position filter - NOT IMPLEMENTED - can be implemented
        in the python source, normally not needed",
        horizontalUnit, (darkSquareHeight + 30) * verticalUnit);
72      controlP5.controller("positionFilterInfoText").moveTo(
        "fineTune");
73
74      pfa = controlP5.addToggle("positionFilterActive", false,
         horizontalUnit, (darkSquareHeight + 31) * verticalUnit,
         20, 20);
75      controlP5.controller("positionFilterActive").setVisible(
        true);
76      controlP5.controller("positionFilterActive").
        setCaptionLabel("Filter position");
77      controlP5.controller("positionFilterActive").moveTo(
        "fineTune");
78
79      // triangle algorithm settings
80
81      controlP5.addTextlabel("algorithmFilterInfoText",
        "Change the properties of the triangle detection
        algorithm", horizontalUnit, (darkSquareHeight + 35) *
        verticalUnit);
82      controlP5.controller("algorithmFilterInfoText").moveTo(
        "fineTune");
83
84      controlP5.addSlider("tolerance", 0.0F, 0.01F, 0.005F,
        horizontalUnit, (darkSquareHeight + 36) * verticalUnit,
        400, 20);
85      controlP5.controller("tolerance").setCaptionLabel(
        "Tolerance");
86      controlP5.controller("tolerance").captionLabel().
        toUpperCase(false);
87      controlP5.controller("tolerance").moveTo("fineTune");
88
89      controlP5.addSlider("neededVotes", 0, 10, 128,
        horizontalUnit, (darkSquareHeight + 38) * verticalUnit,
```

fineTuneTab.pde

```
        400, 20);
 90     controlP5.controller("neededVotes").setCaptionLabel(
        "Needed votes");
 91     controlP5.controller("neededVotes").captionLabel().
        toUpperCase(false);
 92     controlP5.controller("neededVotes").moveTo("fineTune");
 93
 94     dv = controlP5.addToggle("debugVotes", false,
        horizontalUnit, (darkSquareHeight + 40) * verticalUnit,
        20, 20);
 95     controlP5.controller("debugVotes").setVisible(true);
 96     controlP5.controller("debugVotes").setCaptionLabel(
        "Print votes for tangibes in the proxy console");
 97     controlP5.controller("debugVotes").moveTo("fineTune");
 98
 99     // screen aspect
100     controlP5.addTextlabel("aspectInfoText", "Adjust the
        aspect ratio of your touch screen - For example 16 / 9
        = 1.777777778 - set value with enter key",
        horizontalUnit, (darkSquareHeight + 44) * verticalUnit);
101     controlP5.controller("aspectInfoText").moveTo("fineTune"
        );
102
103     aspect = controlP5.addTextfield("touchScreenAspect",
        horizontalUnit, (darkSquareHeight + 45) * verticalUnit,
        100, 20);
104     controlP5.controller("touchScreenAspect").setVisible(
        true);
105     aspect.setAutoClear(false);
106     aspect.setInputFilter(ControlP5.FLOAT);
107     aspect.setText("1.777777778");
108     controlP5.controller("touchScreenAspect").
        setCaptionLabel("aspect ratio of the screen");
109     controlP5.controller("touchScreenAspect").moveTo(
        "fineTune");
110
111
112
113     // save settings button
114     controlP5.Button saveFineTuning = controlP5.addButton(
        "saveAlgorithmSettings", 0, horizontalUnit, (
        darkSquareHeight + 49) * verticalUnit, 100 + 8 +
        horizontalUnit, verticalUnit);
115     saveFineTuning.setCaptionLabel("         SAVE TO DISK");
116     saveFineTuning.captionLabel().toUpperCase(false);
117     saveFineTuning.setColorBackground(color(90, 0, 0));
118     controlP5.controller("saveAlgorithmSettings").moveTo(
        "fineTune");
119   }
120
121
```

Declaration in lieu of oath

Last Name: Becker

First Name: Thomas

Date of Birth: 02.02.1983 in Bonn

I herewith declare in lieu of oath that I have composed this thesis without any inadmissible help of a third party and without the use of aids other than those listed.

The data and concepts that have been taken directly or indirectly from other sources have been acknowledged and referenced.

This thesis has not been submitted, wholly or substantially, neither in this country nor abroad for another degree or diploma at any university or institute.

……

(Place, Date) (Signature)