# THE TANGIBLE PROJECT

Using a mediated sketching table, tangible objects and modern web technologies to extend traditional video conferencing

Johan Andersson
Jonas Brood
Patrik Burström
John Ek
Viktor Lindgren
Mattias Lundberg
Jimmy Nyström
Nicklas Nyström
Elias Näslund
Samuel Sjödin
Stefan Sundin
Alexandra Tsampikakis
John Viklund
Karl Öhman

Supervisor: Prof. Peter Parnes

Luleå University of Technology


February 2013

We want to thank Andreas Nilsson[1] for helping us build the workspace setup at LTU, and EIT for sponsoring equipment and our trip to the conference in Delft.

1 http://www.ltu.se/staff/a/andreas

# ABSTRACT

In this report we discuss a web-based video conferencing prototype that we built during a project course at Luleå University of Technology. The prototype extends traditional video conferencing by adding a so called mediated sketching table (also referred to as a "shared workspace") and tangible items. The mediated sketching table provides a way to share a physical area on your table, allowing multiple participants to use normal pen and paper to collaborate in new and intuitive ways. The tangible items are meant to provide new ways to interact with the system, allowing the user to remove the traditional mouse and keyboard in an effort to get a more analog and mediated experience.

Since this project was done in collaboration with the EIT ICT Labs[2], we got the opportunity to go to Delft in the Netherlands and participate at the Mediating Presence conference. At the conference, we demonstrated our prototype and discussed it with the professors and researchers that were present. Some of those discussions are presented in this report.

Some theory behind what is required to have a good mediated experience is also briefly discussed.

2 European Institute of Innovation and Technology, Information and Communication Technologies

# CONTENTS

# 1

# INTRODUCTION

As video conferencing systems become commonplace, we face other issues than just being able to communicate. Issues such as finding common standards and platforms that make things simpler for the end user. There are also social issues such as increased friction and decreased teamwork efficiency. What we really want is a video conferencing system that provides the feeling that the other person is sitting right in front of us and hide the fact that that person might actually be sitting on the other side of the planet. This "feeling" can be defined in a more scientific term as mediated presence. Two examples of things that affect this feeling are eye contact and low latency.

We want to be able to share a common workspace to further emulate sitting at the same table. The shared workspace uses real paper and pens and is filmed and projected down from above. The reason for using real paper and pens is that it is easier for the participant and gives the feeling that you all actually are sitting in the same room and drawing on the same paper instead of sitting across the globe.

The vision of future video conferencing systems has always been that everyone is portrayed in their correct size using either big screens or a projector solution. Small screens where the participants all look in different directions, does not add anything that voice communication is not already providing. TelePresence TX9000 [1] is a solution created by Cisco that addresses this issue. It can be seen in Figure 1.

Microsoft Surface is a product that uses a big multi touch screen as a board and different kinds of wireless protocol to communicate with different devices such as, WiFi and Bluetooth. When an object, for example a camera are touching the surface the photos are automatically transferred to the board and then you can use your fingers to move them around. An example of a system using Surface is Three's company. It is developed at Microsoft in order to explore interaction between more than two people using the Surface together with real-time video and "explore the subtleties of traditional notions of identity, awareness, spatial metaphor, and corporeal embodiments as they relate to three-way collaboration". [21]

Another problem with traditional video conferencing setups is that the users are still being forced to use a mouse and keyboard for some actions. The alternative is to limit the user input options. With this project, we are attempting to remove the need for a mouse and key-

Figure 1: Cisco TelePresence TX9000 One-Row System.

board by moving the functionality they provide to other interactive elements, such as gestures and small tangible devices. The camera that is recording the shared workspace gives us more opportunity for input, like capturing hand gestures or the pressing of virtual buttons. The tangible devices provide a similar new source of input and output, but with the distinct property that they are real objects that can be moved.

The project was initiated by de Greef, T.J, Gullström, C, Handberg, L, Nefs, H.T, and Parnes, P. in a cooperation between Luleå University of Technology (LTU), The royal institute of technology (KTH) and Delft University of Technology (TUD) in February 2012[2]. The project is funded by the European Institute of Innovation and Technology[4].

## 1.1 PROBLEM DESCRIPTION

The problem we set out to solve was to improve the feeling of presence when communicating over video. As a part of this problem, we especially focused on enabling architects to work together over a larger distance. Unlike other businesses where one can open a shared document or work with different files and uploading them to a shared repository, architects wish to work with paper and pencil. We also wanted to make it easy to present and discuss objects on the desk as easily as if the other person was standing next to you.

Finally, we explored how tangible devices can be used to simplify interaction with the system by providing buttons and information displays that do not interfere with what is currently going on on the computer screen.

## 1.2 PROJECT GOALS

These are the goals we intended to fulfill:

- Have a working system
- Have a working shared workspace
- Use tangible devices to make things easier and interact on another level, instead of using mouse and keyboard
- Investigate how the system can enhance mediated presence and thus improve teamwork efficiency

The first three goals are easy to define and measurable. The fourth goal is harder to measure, so during the conference we discussed it with the researchers from the various universities in order to get a better understanding of how well the system performs in this area.

## 1.3 PROJECT REQUIREMENTS

The project was very open and had few restrictions, where the main requirements are that the system should use WebRTC for communication backend, provide a shared workspace and use tangible devices for user interaction. As much as possible was to be done with HTML5 and JavaScript. This was to promote using native browsers without any plugins. The system should also give the users a sense of presence in order to promote the feeling of sitting at the same table and drawing together. The complete system must be easy to use after the initial setup.

## 1.4 PROJECT DELIMITATIONS

We only used functionality that browser manufacturers currently provided. The exception was drivers and other software required for the tangible devices. This delimitation is only made since current browsers do not allow direct access to hardware from the browser. Since the system requires a ceiling-mounted projector and camera together with a big screen and open workspace area, we assume that not every user have the knowledge to be able to set up the system at their own workstations.

## 1.5 RESEARCH QUESTIONS

A big part of this project was the discussion of how to improve the mediated presence, that is the feeling of presence and trust. The simplest way to improve the feeling of presence is to have big screens situated so that it feels like the colleague is sitting across the table from you. Good image and audio quality is also needed. Constructing a system that lets the users project confidence and gain the trust of the other users is more complicated. According to Heath et. al, one issue is obtaining eye contact. This is not automatic since there is an offset between the camera and the person on the screen and the problem is made even more complicated when there is more than two people in the conversation. Using the system has to be smooth and easy enough for it to be aiding in working together instead of being a hindrance.

## 1.6 PROJECT TRANSPARENCY

One thing that our supervisor, Peter Parnes usually requests in all his courses is that the progress of the project should be periodically updated in blog posts. We have therefore during the project published 45 blog posts[1] with updates about the progress. The idea was that anyone should be able to follow the progress of the project, even people with no connection to the university, and be able to understand the blog posts.

Early on in the project we published a gantt chart with our milestones, see Figure 2. We made room to publish five releases during the project — named Arnold, Jason, Sylvester, Dolph and Chuck from the action heroes from The Expendables movie [11] — although we only needed four releases. Fortunately, we did not need to make any major adjustment to this planning.

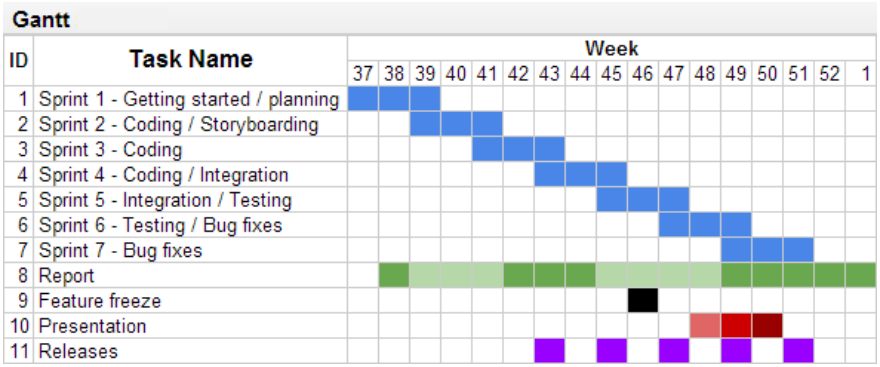


Figure 2: Gantt chart with milestones.

1 http://the-tangibles.blogspot.com

Although not a requirement, we also decided to release all of our code as open source. We genuinely believe that knowledge and academic research should be public and free for all to use and learn from. The code is avaliable at[2].

2 https://github.com/stefansundin/The-Tangibles

# 2

## EXAMPLE SETUP

In Figure 3 below, you can see the basic setup of the system. Here is a brief explanation of its different parts:

1. A screen displaying the chat window is placed at the edge of the table in order to emulate that the other person is sitting at the same table with the workspace between them.
2. The shared mediated workspace. What is drawn inside this area is shared between all users. The keyboard and mouse have been moved aside as they are not the main input devices.
3. The projector is mounted above the workspace.
4. The camera that films the workspace. What is filmed locally is projected on other users' workspaces.
5. A second camera filming the user's upper body. The image data is sent to other users' chat windows.

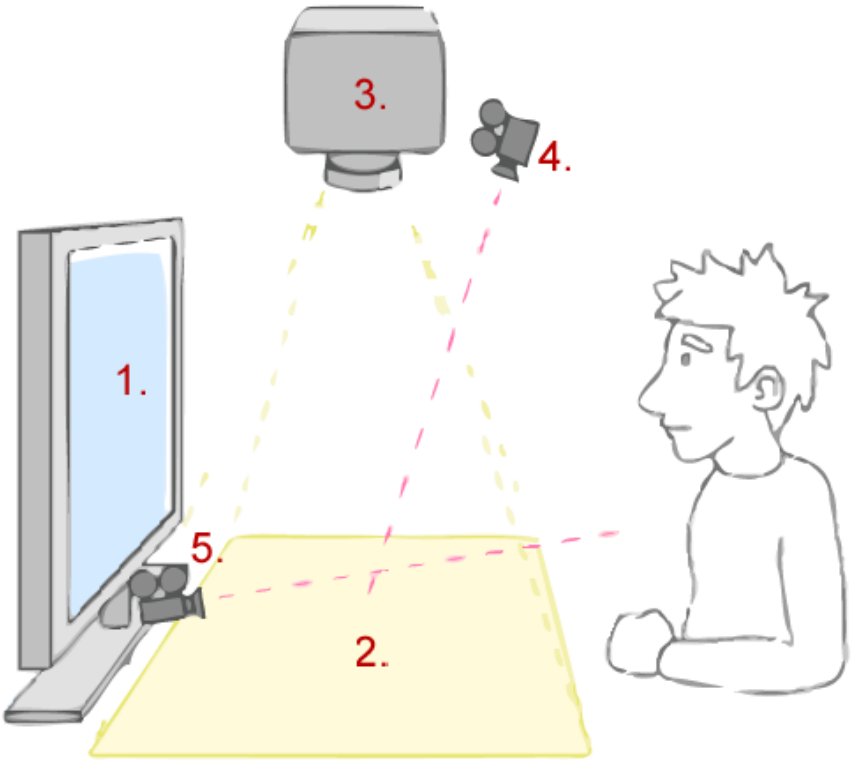


Figure 3: The setup of the system.

# 3 TECHNOLOGIES

This project contains a myriad of different technologies. The main ones are listed in this section.

## 3.1 HTML5 AND WEBRTC

To structure and present information on the World Wide Web, HTML is used extensively where HTML5 [18][23] is the newest HTML standard, succeeding HTML4 and XHTML. The main new features that are interesting for the Tangibles project are the new <video>, <audio> and <canvas> elements.

WebRTC[1] [25] is aiming to become the standard for real-time voice and video communication in the browser, without the use of plugins. Work began in April 2011 and as of December 2012 the standard is not yet completed. A requirement for WebRTC is to use it in conjunction with HTML5. The main idea is to use peer to peer communication instead of sending all the information through a central server. Some sort of server or messaging server is still required in order to set up the connection. This connection, called PeerConnection, is used to transmit the real-time video and voice between the connected clients. The video streams can then be shown using <video> elements.

Our choice to use WebRTC lets us stick to HTML and JavaScript and removes the need for a powerful server, but limits us to use the Google Chrome web browser for the time being.

## 3.2 NODE.JS

Node.js is an event-driven, non-blocking I/O JavaScript framework [14], built on Chrome's V8 JavaScript runtime [9]. The reason for using Node.js is that it makes for easily building fast and scalable network applications. The server is entirely written under the Node.js framework and uses the WebSocket implementation to handle communication to the clients. The other main purpose of the server is to handle room logic and also the simplified API for WebRTC.

1 Web Real-Time Communication

## 3.3 TANGIBLEAPI

TangibleAPI is a gateway that provides a general way of accessing tangible devices connected to your local computer from the web browser. The API is meant to be general, so you don't have to write web applications that targets a specific device, but instead target devices with specific capabilities. The devices have to have specific drivers written for them to expose their functionality to the TangibleAPI server.

The TangibleAPI was developed in a master thesis [13] by Leo Jeusset, which we have continued developing. The original source code was retrieved from Github [12].

## 3.4 SIFTEO

Sifteos are small cubes with a color screen on them [20]. They can be seen in Figure 4 and 5. A wireless connection allows them to interact with the computer. Using the connection, it's possible to show pictures on the screens. In addition to this, they are also able to react to certain events and pass them on to the computer. Some events that are possible is pressing, tilting and placement relative to each other. The driver for Sifteo uses the TangibleAPI to communicate to the web browser and was developed during the same master thesis as the TangibleAPI.

Figure 4: Sifteo cubes.

## 3.5 SPHERO

Spheros [16] are small robot balls. They can be seen in Figures 6 and 7. Spheros can be steered from a smartphone or interacted with from a computer using a Bluetooth connection. The connection has support for reporting movement as well as the ability to send commands such as speed, direction and a color to show by the built-in multi-color lights.


Figure 5: Sifteo cubes.

Figure 6: Sphero ball.

The SpheroDriver [6], developed at LTU by Nicklas Gavelin as part of his master thesis [7], is a port of the Sphero Robotix Android API [17] that enables Sphero to run on a desktop computer. The slightly modified version of this Robotix Android API is here called Sphero-Desktop-API [5]. The Sphero-Desktop-API supports a fairly large amount of commands and functions, such as setting a color, spin, roll, as well as streaming sensor information. The SpheroDriver has a basic foundation for communicating with TangibleAPI, enabling the sphero to be controlled from a website. It was developed in Java with the Bluecove library for Bluetooth communication. Bluecove makes it possible to run in the most common operating systems.

## 3.6 WEBGL

WebGL [10] is a JavaScript API for graphics rendering in the browser. It allows for code to be compiled and run on the computer's graphics

Figure 7: Sphero controlled by a smartphone.

processing unit (GPU), which is more suited for 3D and 2D graphics manipulation.

In short, a WebGL program consists of a vertex shader, a fragment shader and input data, such as vertices of a 3D model and texture images. The vertex shader processes each input vertex, then clipping and rasterization happens behind the scenes, which turns the surfaces of the model into fragments ("potential pixels"). The fragment shader then processes each fragment before the final steps of the graphics pipeline.

The ability to process pixels efficiently and in parallel is well suited for doing the types image processing and transformations that is used for manipulating video streams in this project.

## 3.7 ARUCO

ArUco [22] is a library for augmented reality applications based on OpenCV (Open Source Computer Vision). It is used to detect so called AR markers (Figure 8) in images, each one representing a number between zero and 1023. Several markers can be detected at once, and they can be detected regardless of their orientation.

## 3.8 PIXASTIC

Pixastic [19] is a JavaScript library which provides a wide variety of operations on images. It works by utilizing the access of raw pixel data, which is a feature in the new HTML5 canvas element.

Figure 8: Example of AR markers that can be detected with ArUco.

## 3.9 LOGITECH CAMERA SOFTWARE

Logitech Camera Software [15] allows you to take high quality photos and videos with your webcam, activate motion detection and use face-tracking. It let's you adjust the settings of your webcam in order to improve the image quality. It can be used together with your preferred video-calling software.

## 3.10 YUIDOC

YUIDoc [26] is a documentation tool and a Node.js application. By writing comments in the code in a certain syntax, YUIDoc generates a number of web pages which are easy to navigate through. It is very similar to Javadoc, but works with any programming language that support comment blocks. By using a documentation tool it is easier and faster to get an overview of the system.

## 3.11 GIT

For revision control and code management we use Git. Git is a open source distributed revision control system [8]. To simplify the setup we chose to host the repository at Github[2].

2 https://github.com/stefansundin/The-Tangibles

# 4 USE CASES

These use cases aim to show some real world usages of the application. We have created a few use cases which show the most basic functionality of the system, while more advanced (and out of scope) features, such as actual drawing and design, are intentionally omitted.

The usage of the system is based on three persons. First we have two architects, Alice and Bob, that work in the same company but in different locations. They often work together on various projects and want to save travel time by working over a distance instead. The third person is a project manager, Chuck, who is overseeing this particular project and is located at a third location.

## 4.1 LOGGING IN

When Alice arrives to the workstation in the morning she starts the computer and logs in. After that she starts the web browser and opens the Tangible web page. She is met with the login form shown in Figure 9, where she enters her name and presses continue. She is now logged in and is met by a list of rooms and participants shown in the lobby 10.


Figure 9: Alice logging in to the system.

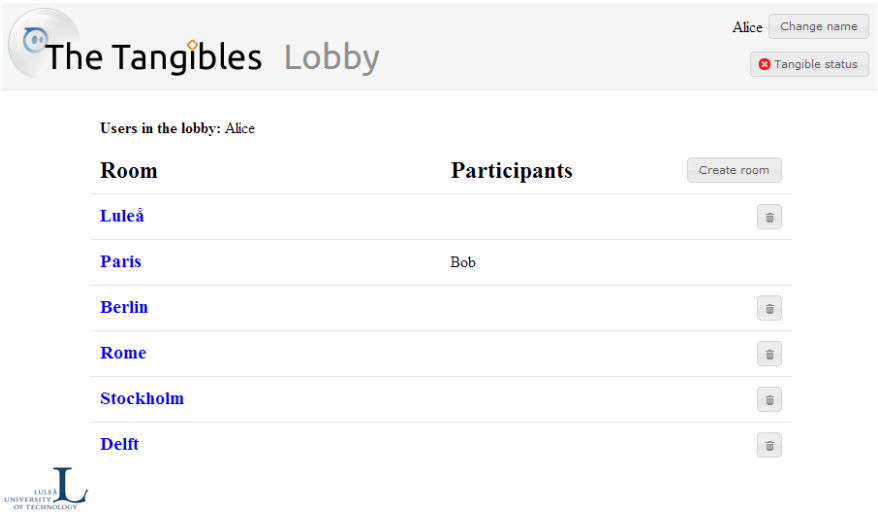


Figure 10: Alice in the lobby.

## 4.2 ENTER A MEETING ROOM

When Alice is in the lobby she sees that Bob is already in the room. She wants to join the room to be able to speak with Bob. To enter the room and speak to Bob she simply presses the row representing the room in the lobby shown in Figure 11.

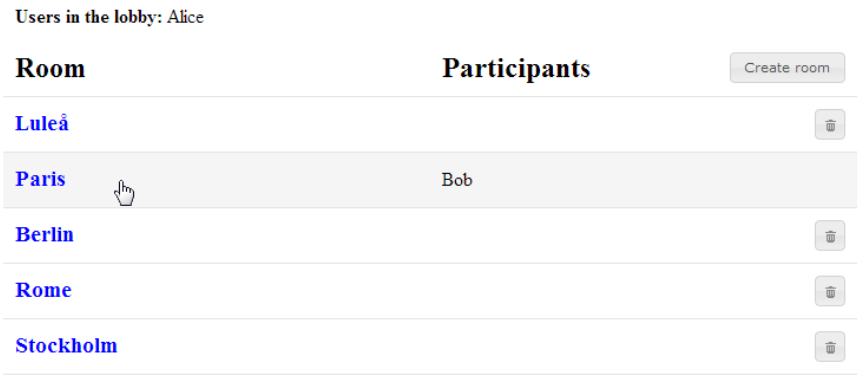


Figure 11: Alice entering the same room as Bob.

Now Alice is inside the room and is met by a video showing Bob. The room view is shown in Figure 12.

## 4.3 USING THE SHARED WORKSPACE

Since Alice and Bob are architects they want to refine their designs. Hence they need to open the shared workspace. Since they have a projector and a camera correctly mounted above the workspace they


Figure 12: Alice entered the same room as Bob.

can just open their workspaces. We will follow Alice, but Bob takes the exact same steps to start sharing the workspace.

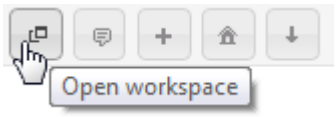


Figure 13: Button that opens the workspace.

Alice begins by starting the projector and making sure the computer recognizes it. She then presses the open workspace-button, see Figure 13. A new browser window opens that is then moved to the correct position. She needs to move it so it is seen on the projected image on the table. She then need to allow the browser to use the camera and select the correct camera filming the workspace. After allowing the camera to be used, Alice can start the automatic calibration with the press of a button. Alice then have the opportunity to adjust the size of the workspace, and when she is happy with the size she just presses the virtual button to the left to continue.

When both Alice and Bob have finished these steps, they are able to see each others shared workspaces. Now they can start refining their designs together.

## 4.4 INVITING SOMEONE

Alice and Bob have discussed the design for a while and decide that they need to discuss some details with their manager, Chuck. Since Chuck is also logged in to the system, Bob invites him. Bob presses the invite button shown in Figure 14. Doing so opens a dialog showing a list of people currently online, seen in Figure 15. He selects Chuck and presses OK. Now Alice and Bob wait for Chuck to accept the invitation.


Figure 14: The invite button.

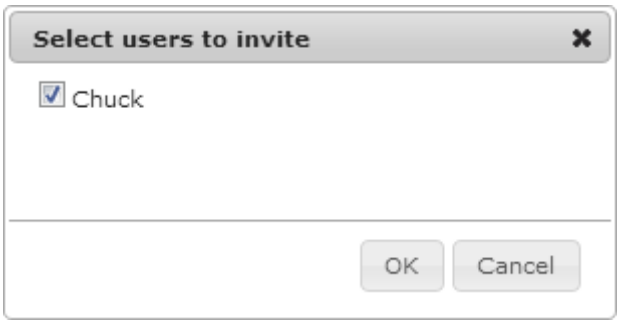


Figure 15: The invite dialog.

On Chuck's end, a notification with the invitation appears, shown in Figure 16. He also sees the notification on his tangible devices, as seen in Figure 17, and can easily accept the invite by pressing the accept button on the web page or interacting with the tangible devices. After he has accepted he will join Alice and Bob in the discussion.

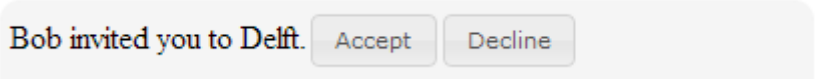


Figure 16: The invite notification on the web page.


Figure 17: The invite notification on the tangible devices.

# 5

# SYSTEM ARCHITECTURE

The completed system consists of various parts. The most important of them are the node.js server that runs separately. Then there is a few different functionalities based in the browser, such as WebRTC communication, functionality for the shared workspace and the tangible web interface. All of those communicate with each other using shared javascript objects.

## 5.1 TANGIBLE DEVICES

Communication from browser to the TangibleAPI are conducted by standard HTTP queries using a RESTful API. Handling the communication this way has several advantages, where the most important is that it does not depend on the use of plugins and works on any browser with good JavaScript support.

The TangibleAPI server also uses a standardized set of messages to communicate with its drivers. The respective device drivers are then responsible to perform the action on the device. An overview of the tangible devices can be seen in Figure 18.

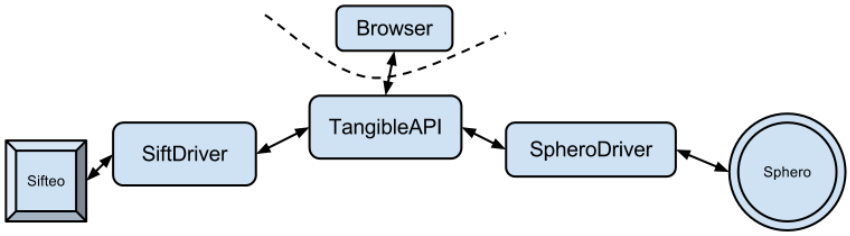


Figure 18: Overview of the communication over the TangibleAPI.

## 5.2 NODE.JS SERVER

The server consists of a Node.js server which opens a WebSocket to each client entering the index page. Using an internally developed API it is possible to communicate between the clients and the server via the WebSockets. The API communication is event driven and is built around packets consisting of one event and a data payload containing information regarding that event. The server is split into

two parts with one handling messages and states regarding clients and another part dedicated to handling communication for setting up WebRTC sessions. The system architecture and communication systems usage is shown in Figure 19.

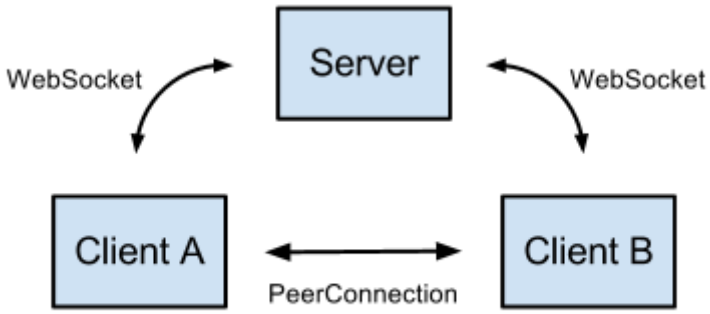


Figure 19: Overview of the system architecture and communication.

# 6 RESULT

This section is divided into separate parts for the different technologies and concepts of the project. First we will discuss the concept of lobby and room and then look at the main technologies. We also present some other things related to the project such as testing and documentation.

## 6.1 LOBBY

When users first enter the site they are prompted to enter a username. After that they are forwarded to the lobby, see Figure 10. This lobby is a portal for the site, allowing users to create new rooms, delete rooms, enter rooms, change their username and also see which users are participating in which room. Users also get notifications about incoming calls at the bottom of the page.

## 6.2 ROOM

A room is the place on the site where users gather up for their conversation and shared workspace. There are no imposed limit on the number of users that may be in a room at once, but due to high resource demand on the computer, only a few users are feasible.

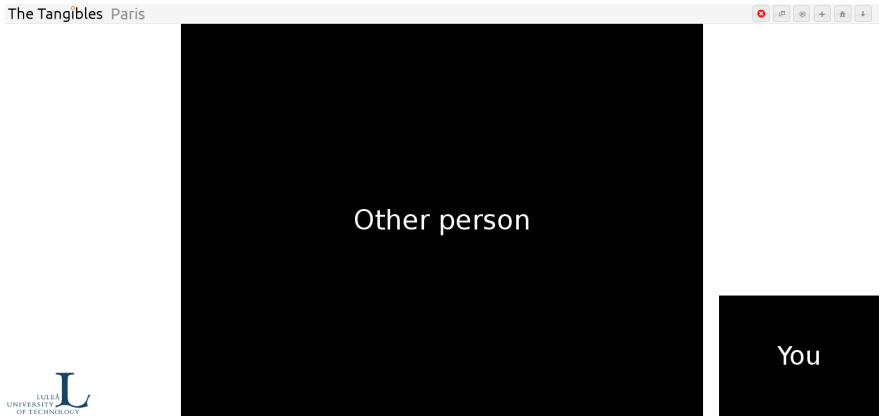


Figure 20: A screenshot of a room named *Paris*, with black areas showing where the video streams would be positioned when there are two persons in the room.

In the center of the page there are video streams of the other participants. A small video of yourself is shown in the bottom right. In the upper right corner there are various buttons to control the site. The buttons are as follows; view the status of the tangible devices, toggle the workspace window, toggle the text chat, invite more users to the room, leave the room, and minimize/maximize the header, maximizing will also allow the user to change their name. The buttons are shown in Figure 20 in that order. To start sharing the workspace the user presses the open workspace-button, moves the new window to the desired position on the projected image, and finally presses a button to start the automated calibration process, described in greater detail in Section 6.3.2.

## 6.3 SHARED WORKSPACE

The mediated sketching table gives users the illusion of drawing on the same physical workspace; they can see each other's drawings and each other's hands. Each user of the shared workspace has a rectangle projected onto the table, the contents of which is shared with the other users. The goal is to make the shared area look as close to identical as possible for all users, giving them the feeling of sitting around the same table. To achieve this, each user records their shared area and streams the video to the other participants.

### 6.3.1 *Setting up a workstation*

The greatest challenge when setting up a workstation is to get the lighting and all the settings just right. There are a number of variables that affect the quality of the captured image, the most important being ambient lighting, projector light intensity and camera settings. The following sections contain instructions and tips for setting up a workstation correctly.

#### 6.3.1.1 *Environment*

- Make sure the windows in the room (if any) has blinds or drapes, to keep the sunlight out.
- Use a light source that gives an even ambient light (we use fluorescent lamps). Directional lighting, such as searchlights or spotlights may work too, but then it's important to position them such that the entire drawing surface is evenly lighted.
- Avoid using glossy paper, as this introduces unnecessary glare that makes it harder to capture the drawings on camera.

- Use broad marker pens when drawing. Internet bandwidth may limit the resolution of incoming video streams, which can make thinner lines disappear.

#### 6.3.1.2 *Camera*

- Mount the camera as close as possible to the drawing surface, just make sure the entire projected image is visible to the camera.
- Use a camera that allows you to change the exposure time, brightness and autofocus settings. We use Logitech C920 cameras with accompanying software.
- Turn off the camera's autofocus feature and focus the camera on the drawing surface.
- Disable the "RightLight" setting in the camera's control panel. Otherwise, the camera will adjust some settings automatically.
- Increase the camera's exposure time until the flickering lines (discussed in the Issues section) in the captured image disappears.
- To reduce the effects of video feedback and to avoid a constantly darkening image, increase the brightness setting on the camera to around 75
- Make sure the gain setting is turned down low. Sometimes, for reasons we haven't been able to deduce, this setting changes by itself when the camera starts.
- The correct camera settings depend on the lighting conditions and choice of projector, so there will always be some trial and error to get it right.
- Make sure you save the camera settings before closing the control panel!

#### 6.3.1.3 *Projector*

- Mount the projector relatively close to the table, about 1.5 meters.
- Turn down the brightness and contrast on the projectors as low as possible while still keeping the projected image clearly visible to the eye. Remember, you want to film as little as possible of the projected image to avoid video feedback problems, but still enough to make the projected AR markers visible to the camera.

- As was stated in the Issues section of the report, it might be a good idea to use an LCD projector. An LCD projector redraws the entire image simultaneously, which may make it easier to film the projection without having to care about colored stripes.

### 6.3.2 *Camera / Projector Calibration*

A lot of work was put into making the camera/projector calibration process quick and intuitive. The focus of this part of the project has been to eliminate the need to position the camera and projector at perfect angles, and instead solve the calibration problems in software.

Say the workspace area a user wishes to share looks like Figure 21.

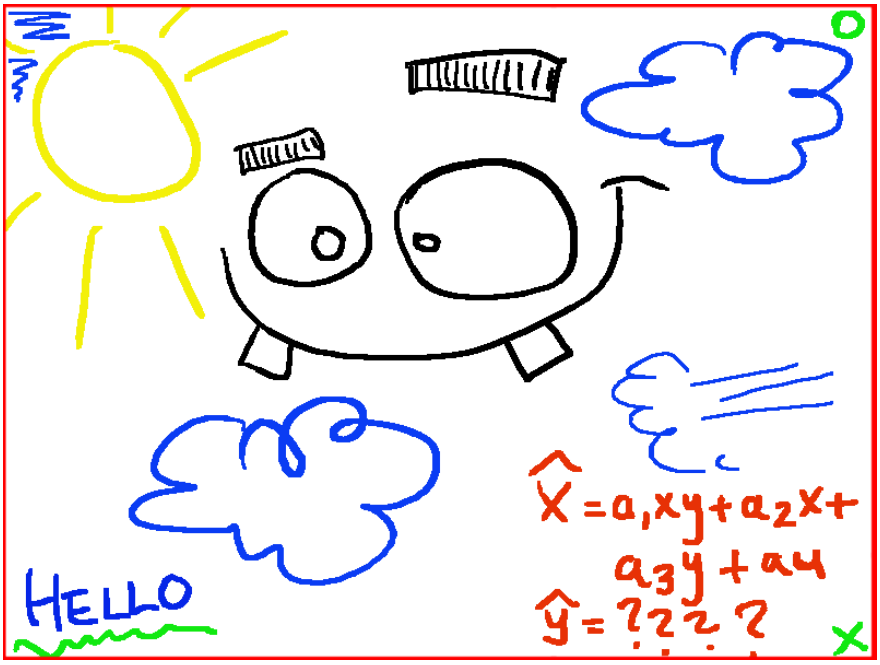


Figure 21: Drawings on a user's local workspace.

When filmed by the camera, the workspace may look something like Figure 22. The other users need some way to transform the distorted rectangle in the video stream to fit into their own shared workspace area.

One way of doing this would be to mount the camera at a perfect angle to only film the shared area, then project the video straight onto the others' shared areas. However, this is very hard, sometimes impossible, to achieve in practice. Another approach would be to calculate a perspective matrix based on how the shared area looks through the camera, then draw the image on a 3D plane that reverses

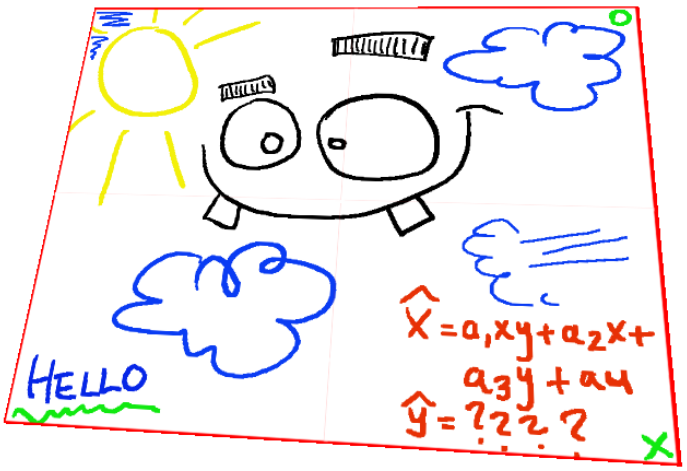


Figure 22: The workspace, as seen by the camera.

the distortion. But unless the projector is perfectly mounted and calibrated, the image will also be skewed by *the keystone effect*, which occurs when the projector is not perpendicular to the surface it is projecting onto. Mismatching resolution between projector and computer can also introduce distortion.

A function is needed to transform an arbitrary point in camera space to its corresponding point in browser window space. A general equation system to convert points from one four sided polygon to another is shown in equation 1.

$$\begin{aligned} x'_n &= a_1 \cdot x_n \cdot y_n + a_2 \cdot x_n + a_3 \cdot y_n + a_4, \\ y'_n &= b_1 \cdot x_n \cdot y_n + b_2 \cdot x_n + b_3 \cdot y_n + b_4. \end{aligned} \tag{1}$$

In Figure 23 an example of two rectangles are shown. When attempting to solve the equations, four points of the shared workspace area are needed in both coordinate systems. We solve the equations and find the desired mapping by performing the following steps:

1. Draw an AR marker in the area we wish to share.
2. Project the workspace onto a table.
3. Film the projection.
4. Detect the marker in the video stream (using ArUco) and extract its corners.

5. Solve a system of equations to get values for $a$ and $b$ in the equations above.

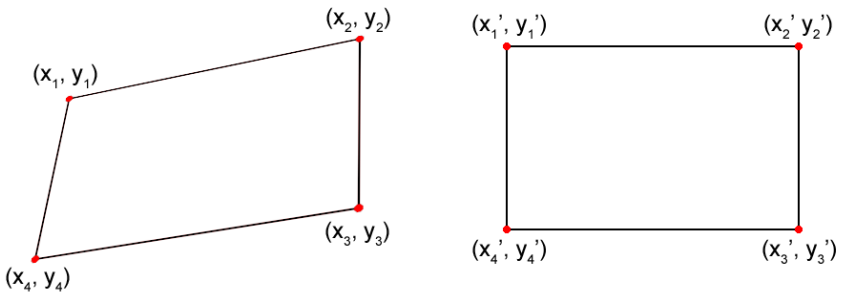


Figure 23: Each point $(x_n, y_n)$ in the camera space is mapped to the corresponding point $(x'_n, y'_n)$ in the browser window space.

A problem with this type of transform is that the resulting image gets distorted, more and more the farther away from the anchor points (corners) you get. If the camera is mounted at a bad angle, this is very noticeable. When the transform is applied to the image in Figure 22 then overlaid on the original image, the difference is clearly visible (Figure 24).

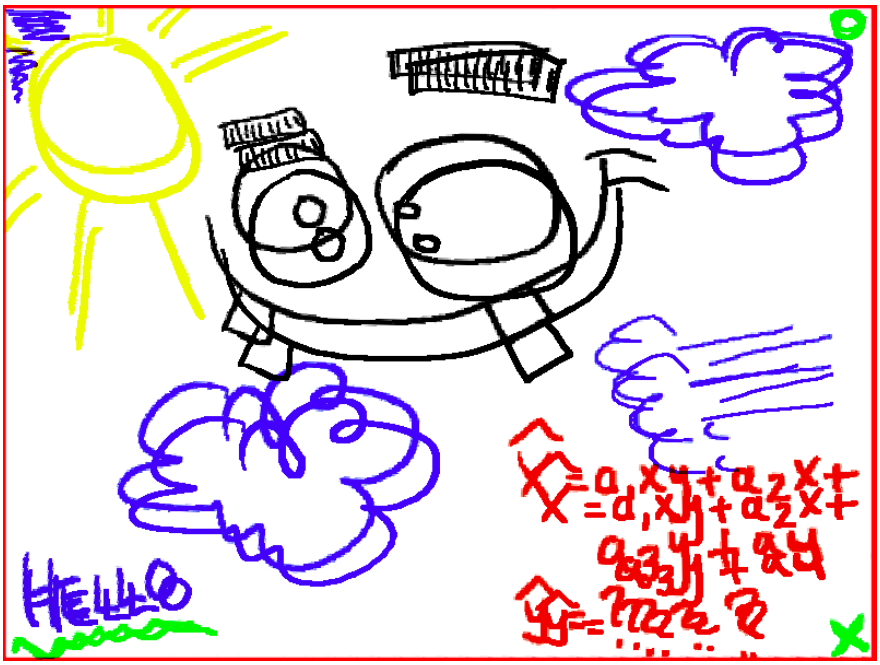


Figure 24: The transformed image overlaid on the original image.

Close to the anchor points the results are acceptable, but near the center the images differ greatly. If the area is split into four regions, as seen in Figure 25, we get more anchor points, and a more precise transform.

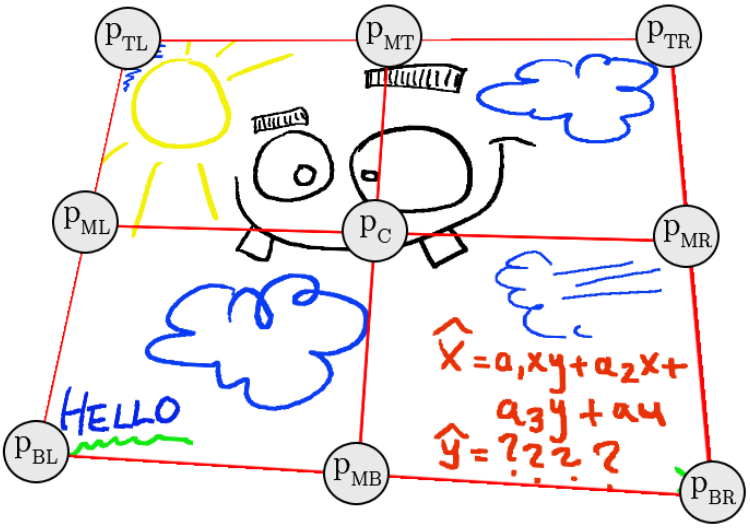


Figure 25: More anchor points used in the transformation (T:Top, M:Middle, B:Bottom, L:Left, C:Center, R:Right).

Applying a separate transform for each of the four regions results in Figure 26. They still don't match up perfectly, but the result is clearly better. Splitting the rectangle into yet smaller regions produces even better results. How we take advantage of this is explained in the calibration steps described below.

#### 6.3.2.1 *Step 1: Finding a transform for the entire browser window*

First an AR marker is projected over the entire browser window, and a transform function is found using the method described above. The reason for doing this is to enable marker and virtual button detection in the following calibration step - when a resizing marker or hand movement is detected, its position needs to be translated into browser coordinates. High precision is not necessary, so one transform is used for the entire area.

#### 6.3.2.2 *Step 2: Resizing and repositioning the workspace*

The outlines of a rectangle are projected onto the users desk. This represents the default workspace area. A virtual button with the text 'Done' is displayed next to it.

The workspace area can now be resized and repositioned by the user. Two Sifteo cubes display separate AR markers. As long as they are seen by the camera, their coordinates (in browser window space) can

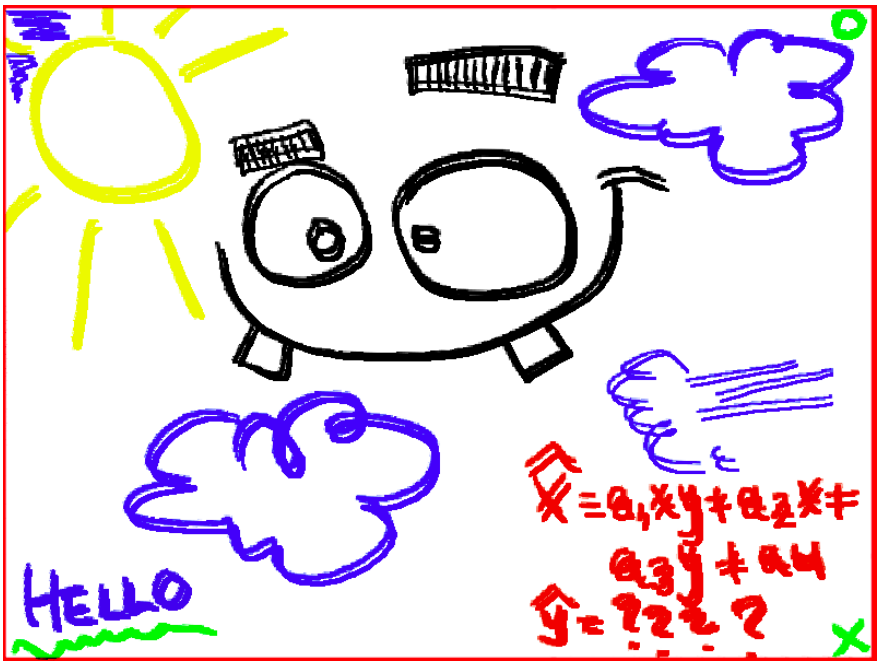


Figure 26: Four transforms applied to the distorted image instead of one.

be calculated using the transform function found in the previous step. One cube is used to decide the position of the projected rectangle, and thus the workspace area: if this cube is detected, the rectangle is moved to its position (the lower left corner of the rectangle is set to the lower left corner of the cube). The second cube is used to decide the size of the rectangle. The width of the rectangle is set to the distance between the cube and the lower left corner of the rectangle. The height is changed proportionally to the width. When the virtual button is pressed — or rather, when motion is detected over the button's projection — this calibration step is completed, and the specified rectangle is used as the workspace area.

#### 6.3.2.3 *Step 3: Finding a transform for the shared workspace*

Next, a function is found to map a point on the workspace in the image data to a point in our computer model.

The solution used is to split the shared area into smaller regions and create separate, simple transforms for each of them. Each transform only manipulates a small part of the video stream, and the results are combined into one image.

The area is split into four subregions, which are then recursively split into four subregions, and so on. The recursion depth is variable and results in $4^{\mathrm{depth}}$ transforms. After some testing, it was decided that a depth of two, giving 16 subregions, was the best choice, as it gives

a good enough result while only prolonging the calibration process a moment.

The mapping found in this final step is sent to all participants of the shared workspace, who then use it to transform the shared area in the corresponding user's video stream to match their shared area in the browser.

## 6.4 VIRTUAL BUTTONS

A virtual button, see Figure 27, is a button that is projected down on the workspace that you can push by moving your hand above it. To be able to push a button a mechanism for detecting movement in the video stream is needed.

This mechanism is implemented by subtracting two consecutive images and see what has changed. By calculating the average value of the three color buffers, R, G and B and subtracting the second image with the first one we get an image showing the difference. Even if nothing has changed between two image captures from the camera stream the images will probably still be slightly different. Therefore we need a threshold to compare the difference to and if the difference is over the threshold we set that pixel to white and if it is lower we set it to black. Now we got a black and white image showing every change in the image between the two captures. If more than a predefined number of pixels have changed, the button is considered to have been pushed.

## 6.5 TANGIBLE DEVICES

The code received has been adapted to better fit our particular application. The different parts of the application will be described separately below.

### 6.5.1 *TangibleAPI*

The TangibleAPI was originally designed with respect to be able to have multiple applications use the same set of tangibles. The programs had to reserve the devices needed and no one else could use them during that time. This was a problem because if the user reloaded the website or if the browser crashed, the devices could not be re-registered. We solved this by allowing objects to be registered even though they appeared to already be in use. Thus if you establish a new session, it will regain control of the API and the devices.

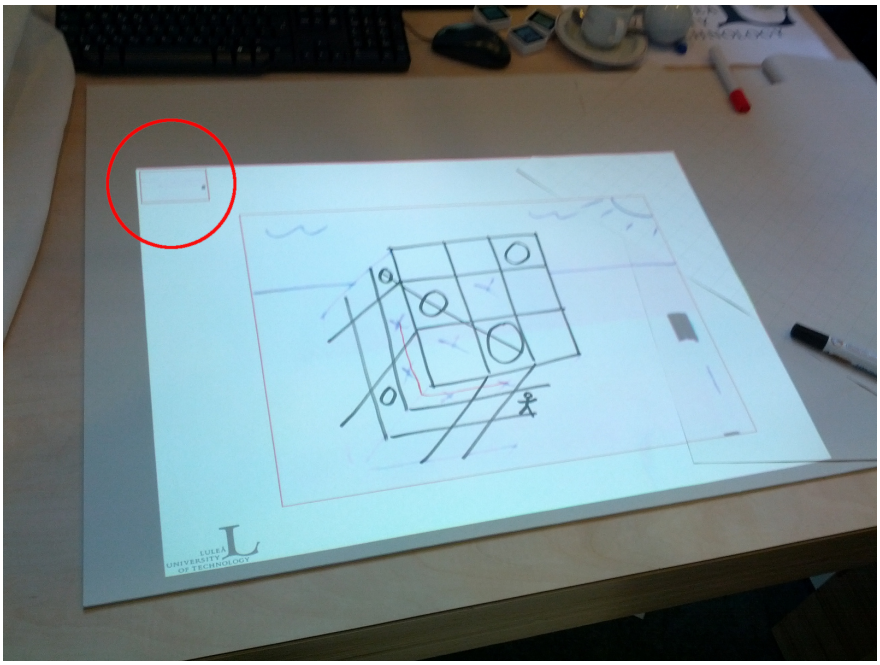

Figure 27: In the upper left corner you can see a virtual button projected down. The button is showing the incoming video stream from another user.

### 6.5.2 *Sifteo*

The Sifteo driver was reasonably well developed and had support for all features we wanted in our application. There were some problems understanding and eliminating some critical bugs during the beginning of the project. After that the driver has mostly worked as intended, although there have been some issues with images. The one that we first had to cope with was the slow transmission and display of images, as it takes several seconds to load and display a new image. To some extent we were able to solve the problem by first showing a text explaining the function before the image is shown. Another problem with images has been that some are not shown correctly, sometimes colors will be wrong. We have not been able to pinpoint and solve this problem, but since we only use static images this problem has not been a problematic issue.

### 6.5.3 *Sphero*

As described in the Technologies section, SpheroDriver already had a basic foundation for communicating with TangibleAPI, enabling the sphero to be controlled from a website. The SpheroDriver has been improved and further developed during this project, enabling new types of commands to be sent from the website in addition to the

previously implemented command `show_color`. The new commands are `spin_left`, `spin_right` and `report_events`. Each new type of event required new code in the TangibleAPI as well as JavaScript for the website, and code for handling these messages in the SpheroDriver. Reporting of events means that a command is sent to the Sphero to initiate a stream, sending either its gyro data or its accelerometer data. The sensor data is sent in full detail all the way to the web page, allowing for building advanced web page applications where the sphero can be used, for example as a controller.

In the use case for this project, when a user is being called/invited by another user to join a room, the sphero starts to spin. The called user may then push the Sphero to give a positive reply and join the room.

## 6.6 TESTING

During development we have had the need to test our code. No automatical tests have been developed but rather been tested manually a bit different for the different parts of the project.

### 6.6.1 *WebRTC*

Continuous testing have been done every week together with the other groups. Most functionality have emerged in an ad hoc manner and therefore specific testing has not been a high priority versus bug fixing and adaptation to changes made in Google Chrome.

### 6.6.2 *Projector*

During the project we set up two stations for testing. This made it possible to find out things like how the lighting in the room impacts on the system, and the optimal distance and settings for camera and projector. It also made it possible to detect any eventual bugs in the system.

Additionally, a number of programs were developed to test the different steps of the calibration process locally, without the need for a projector setup.

### 6.6.3 *Tangibles*

Due to the interactive nature of the tangible devices, it was not possible to develop unit tests to test the full behaviour. Instead a test page was developed, where some of the core functionality could be tested. For example, there were buttons to show images and colors on devices, as well as displaying events from connected objects. This approach made it easy to see and evaluate the expected behaviour. In order to test and calibrate the accelerometer and gyro sensor data streams from the Sphero, a graphical output was created in a test page, see Figure 28.

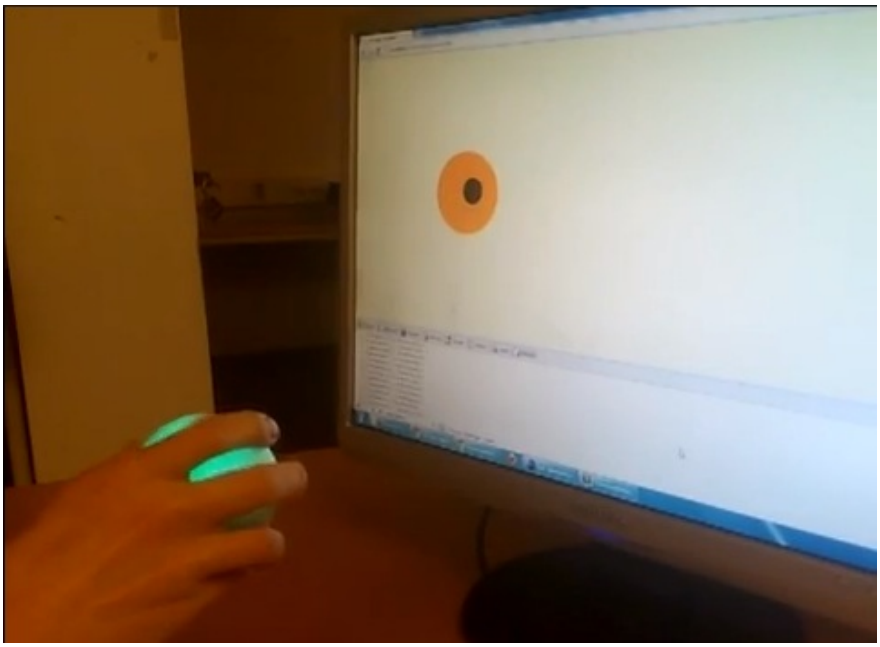

Figure 28: Testing output of Sphero sensor streams.

# 7

# DISCUSSION

During the project we have, as anticipated, had a few problems. In this section we intend to discuss them along with provide some possible solutions. We also intend to provide some examples on how to let follow-up projects continue. Along with those discussions we also present some insight from the mediating presence conference.

## 7.1 ISSUES

The issues are divided by the main technologies used. WebRTC, projector / camera and tangible devices. Apart from those there are no major issues we feel the need to discuss.

### 7.1.1 *WebRTC*

Some problems that have been observed are that WebRTC and Google Chrome are constantly changing, which means that there have been a lot of emergency fixes when the API is changed. Also since the API is constantly changing and not yet finalized, no documentation of the JavaScript API has been made other than the W3C Working Draft [24]. There was also an issue occurring from time to time where the video streams showing the remote participant and the shared workspace switched back and forth every other video frame.

### 7.1.2 *Projector / Camera*

Filming a user's drawing on the table while simultaneously projecting the other users' drawings on top of it can be troublesome. If too much of the projected image is captured by the camera, you run into trouble with video feedback. The other users' images are filmed again and sent back, which may cause lines in the image to grow thicker and thicker if the projected image does not match up perfectly with the original (which it won't, since the calibration process does not produce a perfect result, as previously described). Thus, we want to capture as little as possible of what's coming out of the projector and as much as possible of the physical drawing. However, during

the automatic calibration we rely on the camera to identify projected AR markers, so then we want the camera to see the projected image clearly. To summarize, we want to capture as much as possible during calibration and as little as possible afterwards. One solution would be to run the calibration process once, store the resulting transforms and then change the camera settings to work well when drawing, but we have chosen to compromise.

Our first workstation was set up in a basement lab with no sunlight, which meant that we could control the ambient light however we wanted. This turned out to be more important than we had realized when we tried setting up stations at other locations, with less ideal lighting conditions. At the conference in Delft, for example, huge windows let in large amounts of sunlight, which meant that the lighting conditions changed during the day. This made us to have to constantly adapt the projector and camera settings, just to get an acceptable image. Learning from this, we tried to set up the station at KTH in a completely darkened room. However, with the projector as the main light source, the lighting will change just as much depending on what is being projected (dark images, such as the large AR markers used for workspace calibration, gives much less light than the large white area of the empty workspace). In the end, what is needed to avoid having to change camera and projector settings (after initial calibration) is a room with constant ambient light hitting the drawing area, at least as intense as the light from the projector. More specific instructions on how to achieve this is given in the Setting up a workstation section.

### 7.1.3 *Tangibles*

The Sphero-Desktop-API [5] that is used for PC had some limitation compared to the Sphero driver for Android [17]. One problem was that when the sphero start to spin then there is no way to stop it. A workaround for this was to set it to spin with zero speed. The result is that it stops, but with the side effect that auto balancing is disabled. That is because auto balancing is not possible when spinning due to limitations in the implementation. However, this might be just what one wants when using the Sphero for receiving a call. When pushing at the Sphero, one might not want the Sphero to roll away far but to flip back, staying near its place. Another overall issue with the Sphero-Desktop-API was its lack of documentation and examples. A fairly large amount of time had to be put on understanding how the different commands was to be called upon, this applies especially for the SetStreamCommand, calling the Sphero to stream its sensor data. If the bits or range of integers in the command was a bit off the required range, the command would return nothing. Also no

documentation existed explaining how to reach the returned sensor data, so these two problems both had to be solved by trial-and-error. Another open issue with the streaming command is that although it is possible to command the sphero to stream its accelerometer data or its gyro data, these two can apparently not be sent at the same time. Nevertheless, this has not been a limiting factor for this project.

One of the main issues with the sifteos is the setup phase, where the user is required to use a specific developer edition of the sifteo program. On each run of the sifteo program the user is required to load the driver and press play. This is something that is easy to forget and is not working as expected all the time.

## 7.2 MEDIATING PRESENCE CONFERENCE

The possibility of attending the EIT ICT mediating presence conference [3] at Delft University of Technology was known to us some time in advance. At LTU, due to reasons outside our control, we had only had a few weeks to properly test the setup before departing to Delft. At TU Delft, we had two days to build up a complete setup consisting of two stations. Thanks to a very good effort by the people at TU Delft, we managed to build and test the stations with time to spare. On our third day the conference started, and after a few opening talks, the people attending the conference were able to test the prototype. Since the conference was held in a public space in the university, people who was just passing were also welcome to try the prototype, which many did.

During the conference, we found that the people testing the sketching table didn't need much help in understanding how it worked[1]. Most people thought it was intuitive and games were invented on the spot. This is quite different from playing these games online, as the sketching table is not bound by any rules. This provides the participants the opportunity to agree on their own rules as they play, with both good and less good results. The most popular game was undoubtedly tic-tac-toe. We are very glad that the experience seemed to be intuitive and fun for most people, and it was not uncommon for a session to result in laughter[2].

As part of a scheduled exercise, Robin Schaeverbeke was teaching Jimmy Nyström about perspectives using the sketching table, see Figure 29. Robin went about this exercise as he normally does when his student is sitting right besides him. During this session we gained new insights into how the sketching table can be used as a tool for a

1 Everything was already configured and setup correctly.
2 http://www.youtube.com/watch?v=ehyOn7Ixb1Y

student-teacher relationship. There are some specific technical additions that can be made, such as a viewer-only mode for the sketching table which can be used by the students. Another important part of a student-teacher session is eye contact. The teacher wants to be able to see how the student is responding, while the student might be very focused on the table[3].

The biggest technical problem was that since the sketching table setup is very light sensitive, and the stations were positioned just besides two huge windows, the stations needed to be fine-tuned every once in awhile. Unfortunately, with the current standards, it is not possible to control webcam settings such as exposure time from the browser. If this was possible, some parts of the manual tuning could be automated.

At the conference, the tangible items were mostly unused. Once the session is up and running, the tangible items don't have much use except to simply end the session. This was not too surprising - one of our biggest struggles has been trying to find a really good use case for the tangible devices. A feature that we did want to show in Delft that didn't succeed during those light conditions was using the Sifteos for resizing the shared space. The Sifteo cubes can be used to display the AR markers needed to resize the drawing area for the sketching table. This failed because the screens of the Sifteo cubes have a lot of glare and thus reflect a lot of the light coming from the overhead projector, making it difficult for the camera to recognize the AR markers.

Most of the uses one has of the tangible devices with the current use cases are as easily done with other means such as the virtual buttons, the mouse/keyboard or a piece of paper, but we believe that the Sphero has quite a lot of potential when put in a useful context. One suggestion that was mentioned during a discussion session in Delft was to use the sphero to rotate objects. An assignment was mentioned when the student will get a 3D object from the teacher that he or she is suppose to draw a sketch of. This may seem like something that would be difficult to do remotely, but if the user was holding a Sphero and the model was rotating in the same way the Sphero was rotated, that would make it almost as intuitive as if one was holding the object in the hand. One of the requirements for this is already implemented; streaming the gyro data from the Sphero via TangibleAPI.

3 http://www.youtube.com/watch?v=sWQXCdvuGM4

## 7.3 FUTURE WORK AND IMPROVEMENTS

With the finalization and implementation of the data channels in WebRTC, the project could be expanded to include sending files and documents in an easy way between the clients. However, the current solution will still require a server to find and setup the connections between the clients. Any communication that is only relevant to the participants of a certain room could be sent using the PeerConnection. This would put an even lesser load on the server and limit its work to keeping track of the rooms and performing some work when a user joins a room.

Another feature already mentioned is to use the Sphero to rotate shared 3D objects, for example in an architect teacher's class. It would also be possible to add other tangible devices given that there are use cases included where the device comes to its right.

Multiple improvements on the shared workspace images could be applied, for example subtracting the background or changing all backgrounds to the same color, and enhancing thin lines with WebGL.

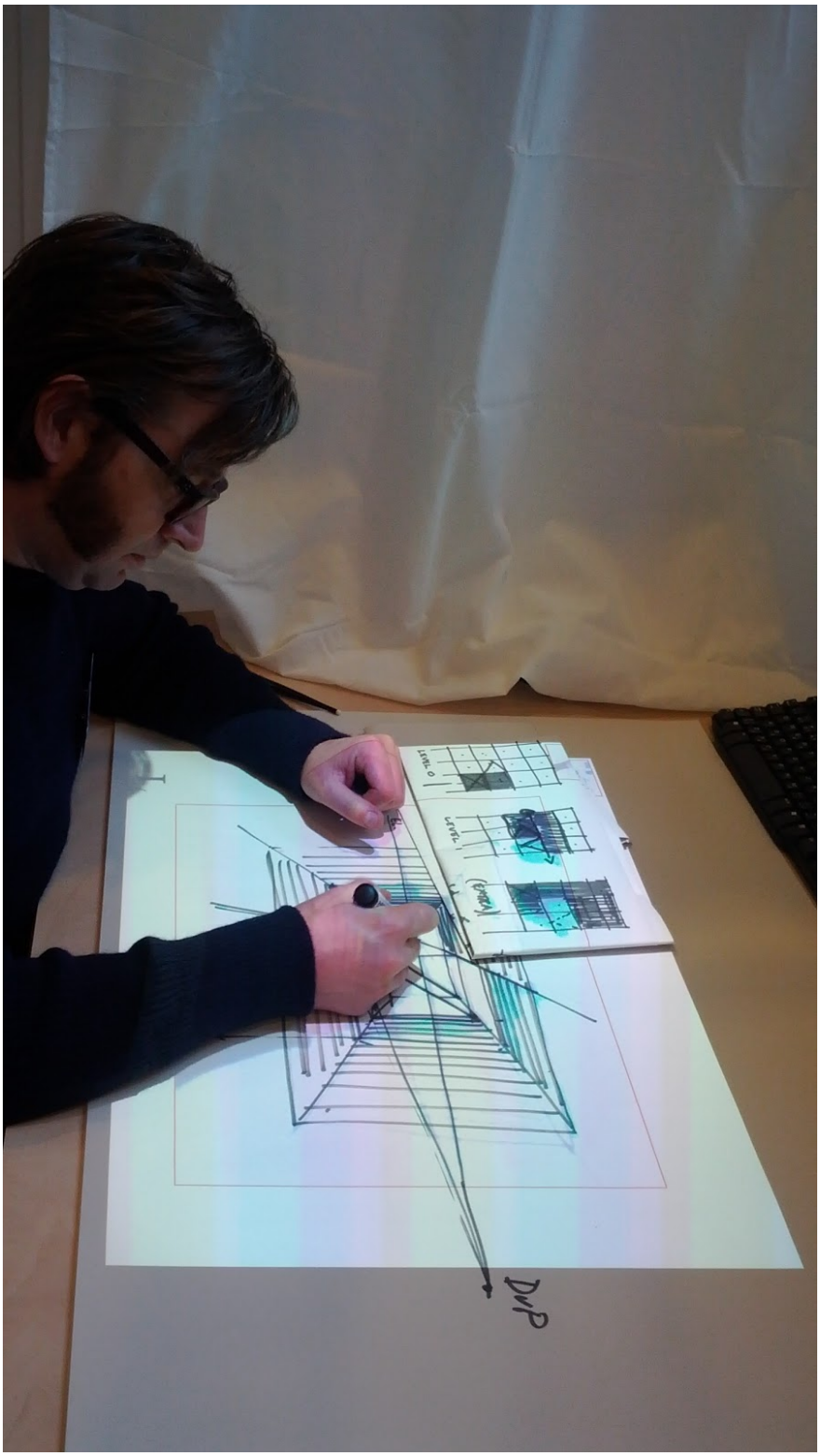


Figure 29: Robin Schaeverbeke teaching Jimmy Nyström using the shared workspace.

# 8

# CONCLUSION

We believe that we have met a majority of the project goals that were set up during the planning phase of the project. We have shown that it is possible and feasible to implement a mediated sketching table using modern web technologies. Since these technologies are changing all the time, we expect that the code will need to be updated with future browser releases for the prototype to continue working.

It would be beneficial to have more low-level access to camera features, such as exposure time and focus control, in order to aid in automatic calibration. But since these web technologies are quite new this is not yet possible. Placing stations in a room with static lighting conditions is currently the best way to deal with this problem.

One problem with the tangible items is that there was no clear use case when we began. It was also very difficult to think of a good use case during the project. Perhaps it was a mistake to start development without one. It should be easy to implement a better use case if development on the prototype is continued.

During the project we visited Delft and demonstrated the prototype at the Mediating Presence conference. We are very thankful that we got the opportunity to go there and we believe it was time well spent. It was challenging to incorporate this trip into the project planning while making sure we had something to show.

A workshop was held near the end of the course at KTH in Stockholm. During that workshop LTU students met with students from KTH and TU Delft to pass on knowledge about how the system works, installing and calibrating workstations, and other things needed to set up a workstation. Stations were set up, early 2013, at each of the universities, and are intended to be used for real video conferencing and applications. The interested reader can find more details about this in Appendix A.

Our hope is that our prototype will be actively used and that these workstations prove to be a valuable tool for those using them.

# A

# APPENDIX: MEDIATED SKETCHING TABLE - WORKSHOP IN STOCKHOLM

What follows is a summary of our experiences during the workshop in Stockholm, 18-19 December 2012.

Jimmy Nyström and Nicklas Nyström from LTU, Henning Alesund from KTH and Arvind Mohabir from TU Delft worked together in setting up two mediated sketching tables at KTH. Professors Charlie Gullström and Leif Handberg assisted with help, guidance and hardware. The first goal of the workshop was to share knowledge on how to set up and calibrate a workstation, since all three universities will have them in the future. The second goal was to find and discuss future work and improvements to the mediated sketching table and finally, experiment with Wacom Boards[1] as an additional means of collaboration.

## A.1 STATION SETUP

The workspace setup was done a bit differently than previous ones. Two flat screen TVs were mounted together and placed standing on a table at approximately a 120 degree angle. The workspace was projected between the screens. The idea was to have similar setups at all three universities, with one person on each screen. This way, all three can look at each other - and see the others look at each other - in a natural way, as Figure 30 shows.

## A.2 REMOTE COMMUNICATION

We got to test using the system for communicating between two different locations (as opposed to having two stations in the same room, as we did at LTU and TU Delft), making this the first time the system was put to the test at actually mediating presence. The station at KTH was used to communicate with Patrik Burström at LTU and collaborate on a drawing 33. When everything is working properly, the system successfully conveys the feeling of working in the same room.

1 A Wacom Tablet is a professional drawing board with an LCD monitor and a pressure sensitive pen that simulates the feel of drawing on paper

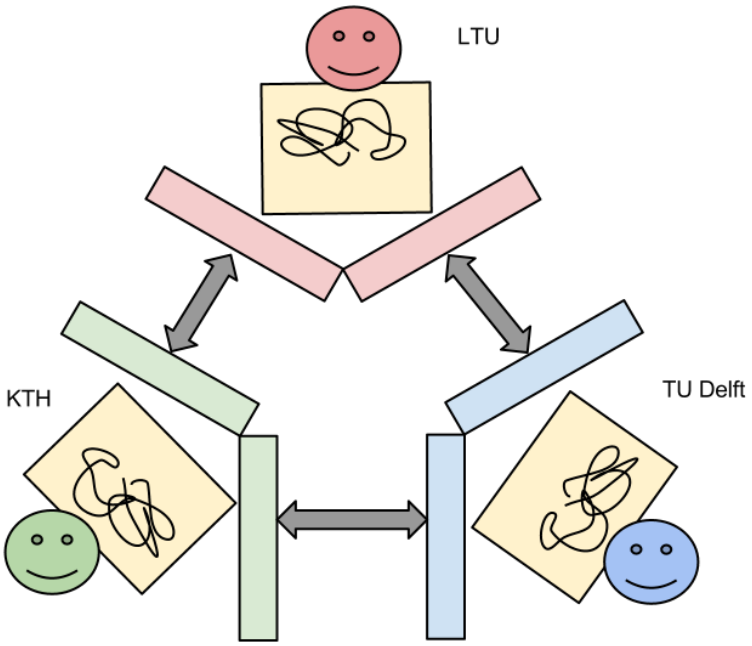


Figure 30: Top view of the three workspace setups, illustrating the idea behind the choice of screen placement.

Due to limitations in the graphics card/drivers we couldn't get the monitors to display the video conferencing in format we wanted, but this can be fixed in software - perhaps as a choice between landscape and portrait mode for the video streams.

### A.3 CAMERA / PROJECTOR CALIBRATION

The station used in LTU had been set up correctly, with a lot of time and effort put into configuring the camera settings. This meant that the LTU station was ready for use right away, with a only a quick and simple camera / projector calibration necessary before starting.

The reason why the LTU station could be calibrated so well was that two stations were set up next to each other. This way, you immediately see the result on the other station as the camera and projector settings are altered. However, in a real world setting, you don't want the users to have to tinker with the camera settings each time they start a session. It should only have to be done once and it needs to be possible to do it locally, before connecting to other users. To achieve this functionality on a single station, new software needs to be written that allows users to directly see the results of changing the camera settings and that also provides help on how to achieve the best results. Without such help, getting everything to work involves a lot of trial and error, and lots of patience.

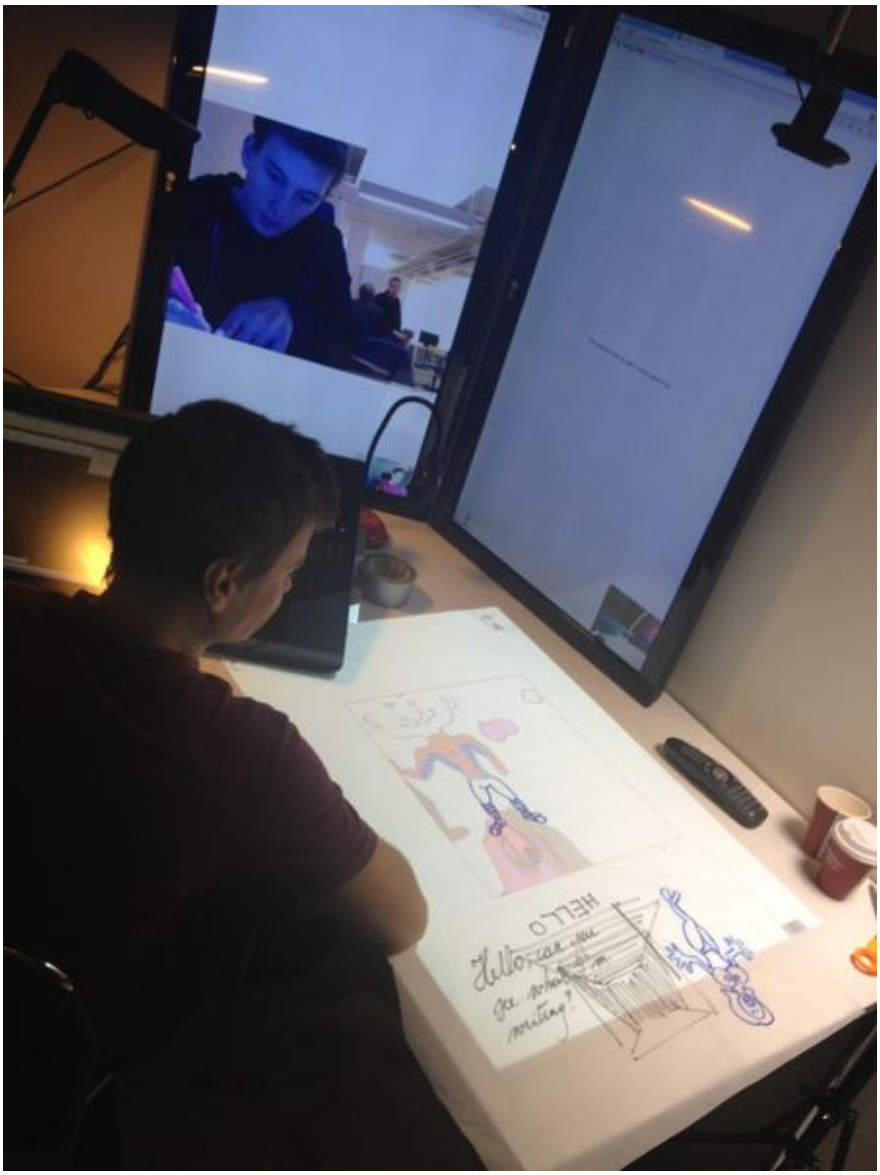


Figure 31: Nicklas and Patrik collaborating on a nice Spider-Man drawing.

### A.4 WACOM BOARDS

We noticed that using the Wacom boards feels a lot like drawing on paper, thanks to their pressure sensitivity. This means that they might be a good replacement to our pen and paper approach. However, the importance of seeing the other user's hands has yet to be determined: during the tests with our system, both users used hand gestures as a complement to the voice chat communication.

As we saw in Delft (the teacher-student experiment), users tend to focus on the drawing itself and may not make much eye contact while drawing, which also points toward the conclusion that seeing the other person's hands is useful.

With the Wacom board, the only indication of what the other user is doing is a little marker (like a mouse pointer) on the screen. This may be sufficient for collaboration, but doesn't really give a sense of presence. As an experiment, the physical pen and paper workspace of the Tangible system was replaced with a Wacom board. The board and camera were placed such that the hands of the remote user was visible on the video chat screen, to give a better sense of what he/she was doing. AutoCAD WS[2] was used as a means for online collaboration with the Wacom boards. It's not made for sketching, but since the workshop was only for two days, we used what was available.

Figure 32: Arvind (left), Patrik (on screen) and Henning (right) collaborating on sketches.

The Wacom board has a lot of potential as a complement to the pen and paper drawing surface, or even as a replacement. However, there are a number of issues that need to be resolved. Firstly, there is no currently available software (that we could find) that supports the pressure sensitive sketching that makes the board great. Secondly, there are many advantages to being able to see the hands of the other users when collaborating, which you don't get when using the board. Using the video conferencing of the Tangible system helps somewhat, but you still lose the ability to gesture and point at the image to convey information. The experiment led to discussions about different solutions to getting the hands to show on the board. One suggestion was to use an approach similar to the shared workspace. That is, to use a projector and camera to simply project video of the hands onto the Wacom board (though this might not be feasible due to glare, among other things). Another idea was to make a transparent video

2 https://www.autocadws.com/

overlay in software, but that possibility needs to be investigated further.

## A.5 HAND PROJECTION POSITIONING

Another question arose as well: what is the optimal position of the projected hands in relation to the chat window?

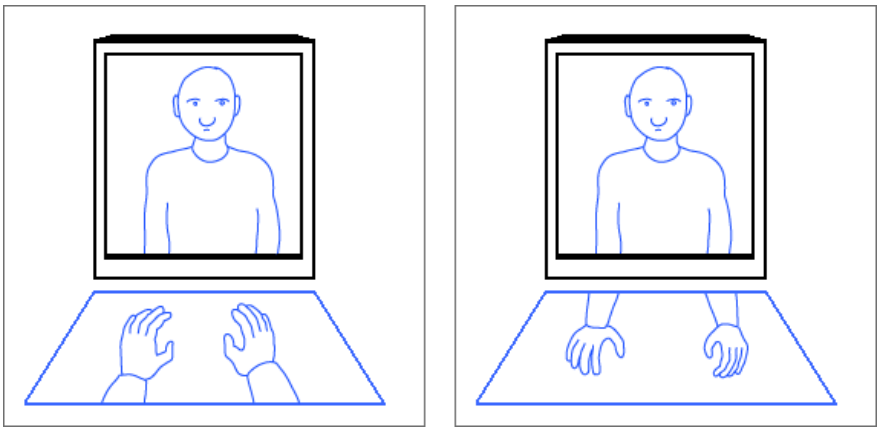

Figure 33: Two possible ways of projecting the hands on the shared workspace.

Above is an image illustrating two possible setups. The version to the left is the one that was implemented in the workshop: when using the system, the other user's hands are projected as though they are coming from the same direction as your own. The right version looks more natural, but poses a couple of problems.

First of all, it makes it more difficult to collaborate on a drawing, since the users will be seeing it from different directions. Second, this approach does not work when more than two people are using the system at the same time; they can't all be facing each other.

# BIBLIOGRAPHY


[1] Cisco. Telepresence tx9000, October 2012. URL http://www.cisco.com/web/telepresence/products/tx9000.html.

[2] T.J de Greef, C Gullström, L Handberg, H.T Nefs, and P. Parnes. Shared mediated workspaces, November 2012. URL http://presencelive.info/Resources/Documents/Proceedings/09_deGreefEtAl.pdf.

[3] Dr. Caroline Nevejan Dr. Charlie Gullström. Mediating presence - workshop and final conference 2012, November 2012. URL http://www.eitictlabs.eu/ict-labs/all-events/article/mediating-presence-workshop-and-final-conference-2012/.

[4] EIT. Eit itc labs, November 2012. URL http://http://www.eitictlabs.eu/.

[5] Nicklas Gavelin. Sphero desktop api, November 2012. URL https://github.com/nicklasgav/Sphero-Desktop-API.

[6] Nicklas Gavelin. Sphero tangible driver, November 2012. URL https://github.com/nicklasgav/SpheroTangibleDriver.

[7] Nicklas Gavelin. Visualizing energy consumption using physical artifacts, November 2012. URL http://pure.ltu.se/portal/sv/studentthesis/visualizing-energy-consumption-using-physical-artifacts%283c3b3195-beef-4a18-a3c1-9adafddf5d97%29.html.

[8] Git. Git, January 2013. URL http://git-scm.com/.

[9] Google. V8 javascript engine, October 2012. URL http://code.google.com/p/v8/.

[10] Khronos Group. Webgl, December 2012. URL http://www.khronos.org/webgl/.

[11] IMDb. The expendables, October 2012. URL http://www.imdb.com/title/tt1320253/.

[12] Leo Jeusset. Sifthesis, November 2012. URL https://github.com/lojeuv/.

[13] Leo Jeusset. Sifthesis, November 2012. URL https://sites.google.com/site/sifthesis/documents.

[14] Joyent. Node js, October 2012. URL http://nodejs.org/.

[15] Logitech. Logitech webcam software, October 2012. URL http://www.logitech.com/en-us/support/5798?osid=14&bit=64.

[16] Orbotix. Sphero, November 2012. URL http://www.gosphero.com/.

[17] Orbotix. Sphero driver, code for android, October 2012. URL https://github.com/orbotix/Sphero-Android-SDK.

[18] HTML 5 Rocks. Html5 rocks, October 2012. URL http://www.html5rocks.com/.

[19] Jacob Seidelin. Pixastic: Javascript image processing, January 2009. URL http://www.pixastic.com/.

[20] Sifteo. Sifteo, September 2012. URL www.sifteo.com.

[21] Anthony Tang, Michel Pahud, Kori Inkpen, Hrvoje Benko, John C. Tang, and Bill Buxton. Three's company: Understanding communication channels in three-way distributed collaboration. *CSCW 2010*, 2010.

[22] Cordoba university. Aruco, December 2012. URL http://www.uco.es/investiga/grupos/ava/node/26.

[23] W3C. Html5 working group, October 2012. URL http://www.w3.org/html/wg/.

[24] W3C. Working draft for webrtc, November 2012. URL http://www.w3.org/TR/2012/WD-webrtc-20120821/.

[25] W3C. Webrtc, October 2012. URL http://www.w3.org/TR/webrtc/.

[26] YUI. Yuidoc, January 2013. URL http://yui.github.com/yuidoc/.