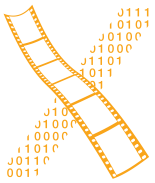


Chair for Computer Science 10 (Media Computing and Human-Computer Interaction)

RWTH AACHEN UNIVERSITY

Understanding How Tangibles Are Used for Data Sharing

Master's Thesis
submitted to the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Asif Mayilli

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Ulrik Schroeder

Registration date: 04.08.2019
Submission date: 28.01.2020

Eidesstattliche Versicherung

_____ _____

Name, Vorname Matrikelnummer

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

_____

_____

_____

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

_____ _____

Ort, Datum Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

_____ _____

Ort, Datum Unterschrift

Contents

List of Figures

Abstract

Tangible User Interfaces (TUI) attract a wide range of users. They rely on the human instinct to be involved and inventive and can provide a means of communicating with computer applications in ways that exploit the awareness and interaction abilities of users with the real, non-digital world. Therefore, more and more research is done in this area. [Underkoffler and Ishii, 1999] Tangibles increase speed, accuracy, and awareness of users. Tangibles are often used with large touch screens called tabletops. The large screen size of tabletops makes it possible for multiple users to collaborate. Also, touch screen surfaces will become cheaper to produce, which will make them affordable.

Collaborative work on tabletops involves *data sharing* which can be performed with tangibles. Therefore, we decided to investigate how people share data using tangibles while working on a tabletop.

In this thesis, we introduce TangiFlow, a tangible-based, general-purpose visual programming tool. We also used this tool to examine how people share data using tangibles. We have discovered two new interaction techniques for tangible data sharing in addition to 2 existing ones. Furthermore, we found out why people sometimes do not share tangibles during data sharing tasks and how the increase in the amount of tangibles affects the process of data sharing.

Acknowledgements

First of all I would like to thank Krishna Subramanian, M.Sc., for being my supervisor. I am grateful for your time you have invested in me guiding me throughout the thesis.

I would like to thank my thesis advisor, Prof. Dr. Jan Borchers, and second examiner, Prof. Dr. Ulrik Schroeder for their time and support.

Thanks to all those who had time to participate in my user studies and gave me valuable feedback.

I want to thank all who have given me moral support during my studies. I want to thank my family for their moral and material help. I am who I am thanks to you. Thanks to my friends who are like family to me, both near and far. Even when I was down you believed in me and supported me in the worst of times for me. You have never left my side and always motivated me and that is what helped me to finish my studies.

Finally I want to thank me for never quiting and slowly but patiently moving towards my goal.

Live long and prosper!

Asif Mayilli.

Conventions

Throughout this thesis we use the following conventions.

Text conventions

Definitions of technical terms or short excursus are set off in coloured boxes.

EXCURSUS:
Excursus are detailed discussions of a particular point in a book, usually in an appendix, or digressions in a written text.

Definition:
Excursus

Source code and implementation symbols are written in typewriter-style text.

```
myClass
```

The whole thesis is written in American English.

Download links are set off in coloured boxes.

File: myFile[a]

[a] http://hci.rwth-aachen.de/public/folder/file_number.file

Chapter 1

Introduction

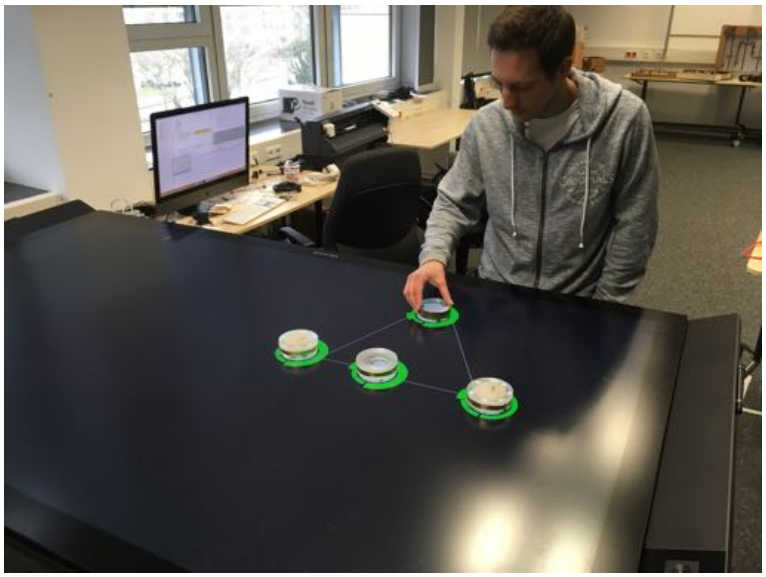


Figure 1.1: The Microsoft Surface Hub[1]

Tabletops provide sufficient space for 2-4 people to gather around them and to work together on them.

Large screen computers that support multi-user interaction are called tabletops. Nowadays tabletops are widely used in different situations and environments, e.g., in a museum where users can learn more about history or art. Their screens provide sufficient space for 2-4 people to gather around them and to work together on them. Since users manipulate data on tabletops directly via touch gestures,

[1] https://www.microsoft.com/en-us/surface/business/surface-hub-2

they are considered more natural than conventional desktop WIMP-based GUIs.

Tabletops are often used with tangibles and together they form a TUI.

Tabletops are often used with **tangibles**, physical devices that can represent or assist to manipulate digital on-screen data. Together they form **Tangible User Interface (TUI)**. TUIs are used in different areas, e.g., in music creation (Reactables [Jordà et al., 2006]), urban planning (Urp [Underkoffler and Ishii, 1999]), programming (Algoblock [Suzuki and Kato, 1995]). There are several benefits of TUIs such as:

- They support *collaborative* interaction [Shaer and Hornecker, 2010].
- They have a *welcoming* environment for expert and novice users [Shaer and Hornecker, 2010].
- They provide *haptic feedback* which is missing during touch interaction [Shaer and Hornecker, 2010].
- They make it possible to work *eyes-free* and be *more involved* with other participants instead of looking constantly at screen [Shaer and Hornecker, 2010].

Figure 1.2: Cooper Hewitt Smithsonian Design Museum touch screen tables[2]

One of the areas where TUI is used is programming.

As we mentioned TUI is also used in the area of programming. For the last four decades, different **Visual Programming Languages (VPL)** based on TUI were developed. VPL is any programming language that allows users to program using graphical elements or their physical representations in contrast to conventional textual programming languages. VPLs are used in different areas, e.g. game development (Unreal Engine, Blender), music creation (MAX/MSP, Reactables), data analysis (Orange, Knime, RapidMiner). VPLs are helpful tools for people who are not familiar with conventional programming languages. Furthermore, VPLs also help experts in the programming area to create programs easier and faster. VPL tools are quite powerful and comfortable to use due to several factors:

- VPLs make programming *easier*[Edwards, 1988].
- VPL program can be interpreted while the source code is being written [Lyons et al., 2001].
- Visual information is processed *more effectively* than textual[Whitley, 1997].

A large screen size transforms into a disadvantage when it comes to sharing data, which can be solved with tangibles.

As it was mentioned above, a large screen size of tabletops allows collaborative work between several users and helps to accomplish tasks more efficiently, as people can work in parallel on the same workspace. However, a huge screen size transforms into a disadvantage when it comes to sharing data. Users can remain far away from one another and it turns out to be difficult to keep up information exchange between them. One of the ways of overcoming this issue is using tangibles. With the help of tangibles, users can share data more naturally and much faster.

Subramanian et al. [2007] reviewed digital and tangible data sharing and found out that tangible data sharing is *significantly faster* than digital data sharing. They also pointed out two interaction techniques for tangible data sharing called **handoff** and **deposit**. However, they did not look

[2]https://www.cooperhewitt.org/new-experience/

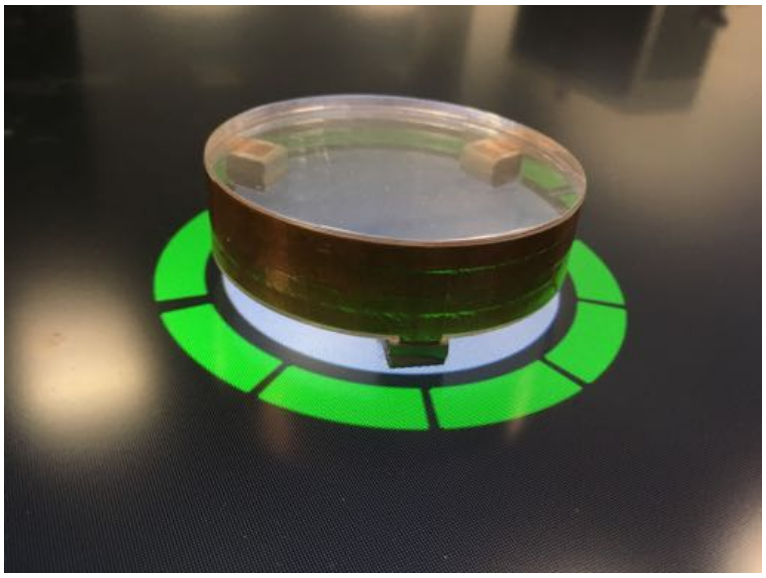


Figure 1.3: Primitive tangible device.

into the behavior of people using tangibles for data sharing and several questions are still open. For example, how the behavior of users change with more tangibles available on the tabletop? Does it make sense to increase the number of tangibles with the increase in the number of users? Do users perform other interaction techniques than handoff and deposit? We looked deeper into these questions in this thesis.

Tangiflow is a tangible-based VPL.

To answer those questions we needed to perform user studies and for this, we needed an environment where we could perform our studies. For this reason, we developed **Tangiflow**. Tangiflow is a tangible-based general-purpose VPL. We used Tangiflow in our studies to validate our research questions. We will talk about Tangiflow in more detail later in this thesis.

Chapter 2

Related Work

This chapter introduces research and existing projects related to passive and active tangible detection, tangible-based data flow VPLs and also data sharing using tangible devices. The first part of this chapter will address a survey of existing VPL tools we have made to find minimal requirements to produce a VPL. In the second section, we will talk about tangible detection research. The third section of this chapter will talk about tangible-based programming tools. The last section of this chapter will address existing research and problem in data sharing using tangibles.

2.1 Survey of Existing VPLs

We have found similarities between different VPLs and used it in TangiFlow

To understand how VPLs function and how to build Tangiflow, we had to analyze lots of VPLs. The main question here was what kind of functionality in terms of interaction we must develop to create a VPL? Lots of desktop-based VPLs were developed so far for different purposes, such as game development (Gamesalad, Blender), music creation (Max/MSP), statistical analysis (Orange, Knime), etc. We have distinguished similarities between all programming tools. They all have similar interaction techniques for similar tasks. In all of them, users can create a node or delete a node, they can connect and disconnect them, etc. A more

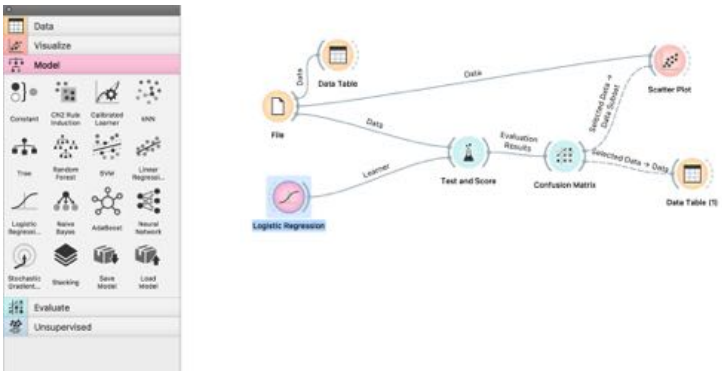


Figure 2.1: The user interface of Orange.

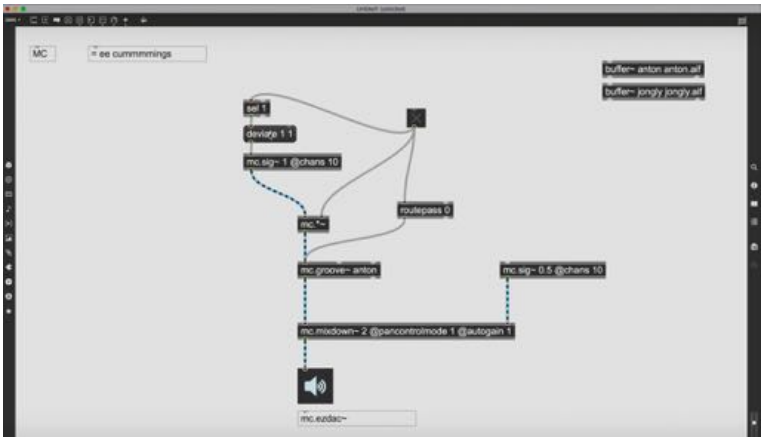


Figure 2.2: The user interface of MAX/MSP.

detailed comparison and the result of the survey can be seen in Figure 2.4.

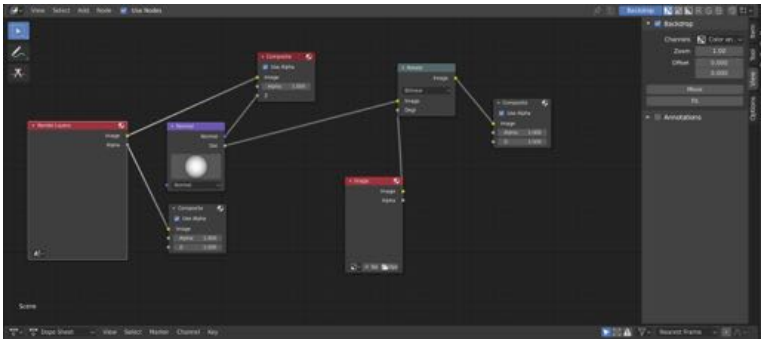


Figure 2.3: The user interface of Blender

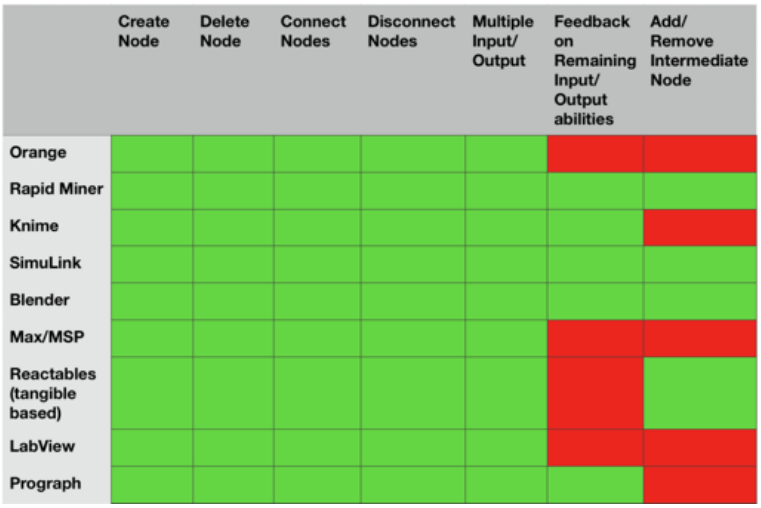


Figure 2.4: Results of the survey. Green color represents the existence of some functionality, and red color shows the absence of some functionality.

2.2 Tangibles

There are two types of tangibles that can be distinguished by capacitive touch screens: passive and active.

Tangiflow is a tangible-based VPL, and we needed to build tangibles that would work on a capacitive touch screen. In the past, most of the tangibles were built for visual multi-touch systems, but those systems require additional devices, such as cameras, projectors, etc. Another disadvantage of such systems is sensitivity to external lighting conditions. Capacitive touch screen tabletops don't have such problems, and thereby nowadays touch screens mostly use capacitive sensors. There are two types of tangibles that can be detected by capacitive touchscreens: active and passive. Passive tangibles are tangibles that don't have active elements of detection, and they rely only on the touch surface of the tangible, whereas active tangibles have active electronic elements, that can detect if tangible is on the surface of the touch screen by sensing electromagnetic signals of the screen. **Passive Untouched Capacitive Widgets (PUCs)** [Voelker et al., 2013] are an example of passive tangibles and **Persistently Trackable Tangibles on Capacitive Multitouch Displays (PERCs)** [Voelker et al., 2015] are an example of active tangibles. Both of these tangibles are supported by **MultiTouchKit (MTK)** [Linden, 2015] frame-

work, about which we will talk later in this thesis. Tangiflow supports PUCs out of the box and with a minor update to the codebase it can also utilize PERCs.

2.2.1 PUCs

PUCs are passive tangibles that use pads and copper foil, which helps touch screens to detect them.

PUCs were introduced by Voelker et al. [2013], and they work on unmodified capacitive touch surfaces. PUCs are made of organic glass, pads and copper foil. Pads create touchpoints on the screen and are connected between each other with copper foil, which creates conductivity. Combined copper foil and pads create touch patterns. The system distinguishes different tangibles on the screen using the pattern of three touchpoints. Before each frame update, those patterns are being identified and assigned to the tangible. Once touch patterns are linked to the tangibles, they get a position and orientation on the screen.

PUCs are cheap to produce, but they have issues with detection reliability.

Among the advantages of PUCs, we can point out the cheap price of manufacturing, availability of components to create tangibles and the lightweight of tangible itself. There are two main limitations of PUCs. First of all, since the uniqueness of tangibles depends on the marker pattern, the amount of uniquely distinguishable tangibles is limited with the size of the bottom part of the tangible. We could have come up with only four different patterns for the given diameter of the base of the tangible. Another limitation of PUCs comes from the working principles of capacitive touch screens. All capacitive touch screens, after a while, filter out persistent touchpoints as noise, so it is hard to distinguish if tangible was taken from the surface or its touchpoints were filtered out by screen. Those limitations were eliminated in PERCs.

2.2.2 PERCs

PERCs are an extension of PUCs.

PERCs are an extension of PUCs. PERCs use the same pattern-based detection, but In contrast to PUCs, PERCs have a field sensor. When the user places tangible on the

screen, the system detects the pattern of touchpoints and field sensor detects touchscreen's electromagnetic signals. Tangible sends information about detected signals via Bluetooth to the system, and then system pairs touch pattern of the tangible and id of the Bluetooth module based on their timestamps.

PERCs don't have limitations of PUCs.

As we mentioned previously, PERCs don't have limitations of PUCs. When touchpoints of tangible disappear, the field sensor can distinguish whether tangible was removed from the surface or was touchpoints just filtered out. Also, the system identifies tangibles using the id of the Bluetooth module, and it is not necessary to have unique patterns. Among the disadvantages of PERCs, we can mention that it is harder to produce them, and they are more expensive to build.

2.3 Tangible Programming

One of the use cases of TUIs is programming. The idea of tangible programming interfaces was around for the last forty years. There were several tangible-based programming tools, such as TORTIS [Perlman, 1974], Algoblock [Suzuki and Kato, 1995], and Reactable [Jordà et al., 2006].

2.3.1 TORTIS

TORTIS is considered the first tangible-based programming language.

It is considered the first tangible-based programming language. This tool allowed children to create physical programs in the Logo programming language. It had two terminals: button boxes and a slot machine. The slot machine terminal was considered a tangible programming tool. Users were able to place cards with commands in slots and move robot turtle with those commands. The main goal of TORTIS was to eliminate obstacles related to typing for children and bring the advantage of learning programming languages to children. It allowed children to manipulate physical objects to create a program, rather than type commands on the computer.

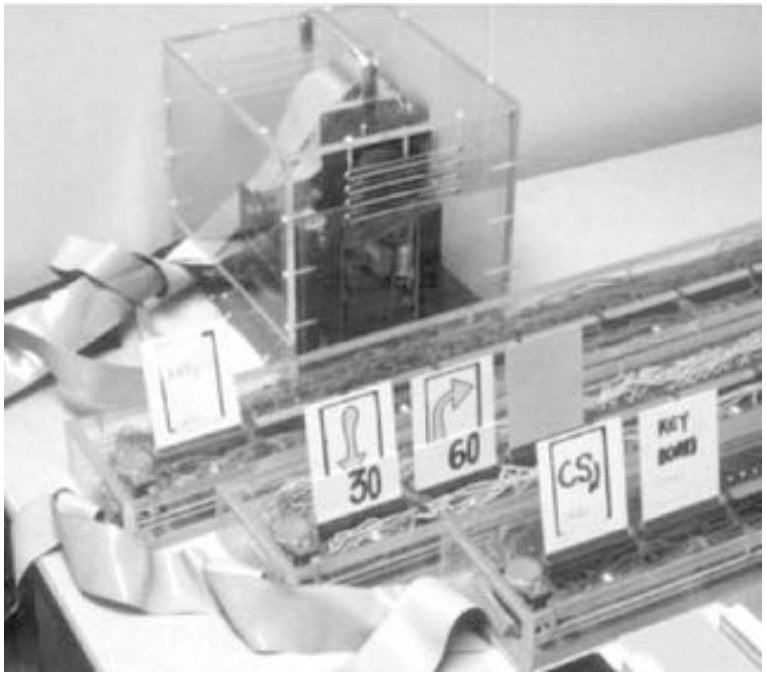


Figure 2.5: Radia Perlman's TORTIS "slot machine"-the part of her system which enables children to write procedures.

2.3.2 Algoblock

Figure 2.6: Algoblock.

Algoblock helps learners to interact and collaborate during solving the problem.

Suzuki and Kato [1995] introduced a tangible programming tool called Algoblock. Like TORTIS, it is a programming

tool for children. Algoblock consists of cubic tangibles, that represent commands like go forward, turn left, turn right, etc. Users can combine blocks to create a program that will help a submarine to reach the destination. The result can be seen on a monitor. Algoblock helps learners to interact and collaborate during solving the problem. In contrast to conventional programming languages, where there are input tools for one user only, which makes it hard to collaborate, Algoblock provides several input tools as tangibles. Users can share tangibles and collaborate. It is quite a primitive tool and does not provide much functionality.

2.3.3 Reactable

Figure 2.7: Reactable user interface.

Reactable is a music-oriented VPL.

A more advanced version of tangible-based visual programming is Reactable. Reactable provides music editing and music creation functionality through connecting tangibles in a graph structure. Tangibles have unique markers that help the camera, which is installed beneath the table to identify and track them. Each tangible represents one music instrument, music source or sound modifier. Connections between tangibles are established by proximity rules. One of the main disadvantages of Reactable is that it is not a general-purpose programming tool. Users are tied to music creation. Furthermore, the expandability of functionality is

another issue of Reactable, users just can not come up with their modules for music editing.

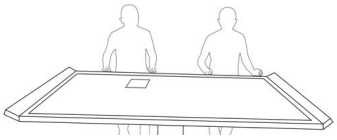
2.4 Tangible Data Sharing

MediaBlocks are one of the early examples of the usage of tangibles as data carriers.

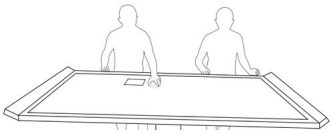
The idea of storing data in a tangible and sharing is not new. An example of tangibles as a data storage can be mediaBlocks [Ullmer et al., 1998]. MediaBlocks are small wooden blocks that can be used as a bridge to share data between different devices or users. MediaBlocks do not store actual data on them but rather play a role of token for data stored online or on other data storage. MediaBlocks can be seen as a physical realization of "copy and paste" functionality. However, there has not been done much research on interaction methods and users' behavior while tangible data sharing in the work of Ullmer et al. [1998].

Do people use other techniques while exchanging data? How does the amount of tangibles affect users?

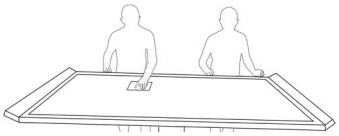
Subramanian et al. [2007] researched tangible data sharing. In their work, Subramanian et al. [2007] compare tangible and digital data sharing. They identified two types of data sharing techniques: handoff and deposit. Handoff is a synchronous action where the sender moves the object or tool towards the receiver and holds it until the receiver takes it. While performing a deposit action, however, the sender does not need to wait for the receiver to take the object. A comparison between tangible and digital handoffs showed that tangible handoffs are performed significantly faster. Authors link this behavior to the reliance of users on well-learned real-world actions to accomplish given tasks. Another finding from this research suggests that the usage of tangibles is not always beneficial. Tabletop systems where collaborative interactions rely on pen and finger input, using tangibles may require additional effort. In Subramanian's study, participants had only one tangible. After each handoff or deposit action, the receiver had to return tangible to the sender. In their work, they did not investigate how the amount of tangibles affects users' behavior, how in general people behave while passing around tangibles. It is not clear whether people utilize other techniques while exchanging data.



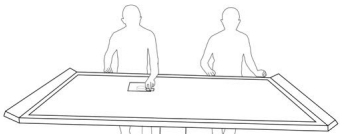
(a) Sender has an image to give to reciever.



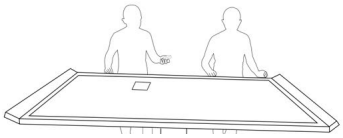
(b) Sender gets a tangible.



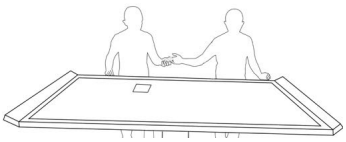
(c) Sender places tangible on the image.



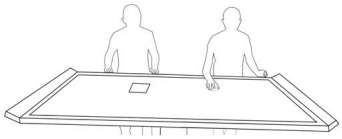
(d) Sender presses copy button.



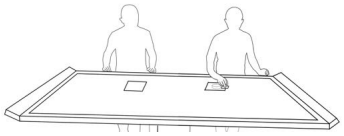
(e) Sender gives tangible to reciever.



(f) Reciever places tangible on his workspace.

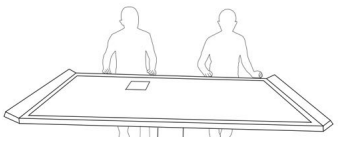


(g) Reciever presses paste button and can use the copy of the image on his workspace.

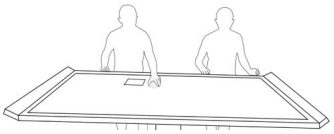


(h) Part 8

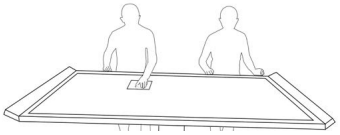
Figure 2.8: Handoff.



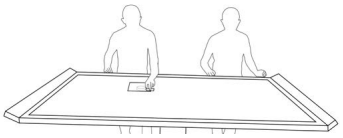
(a) Sender has an image to give to reciever.



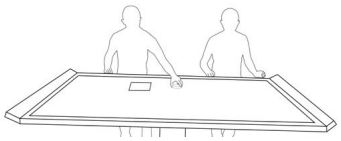
(b) Sender gets a tangible.



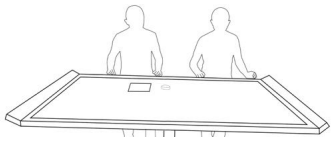
(c) Sender places tangible on the image.



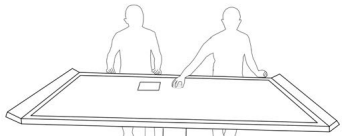
(d) Sender presses copy button.



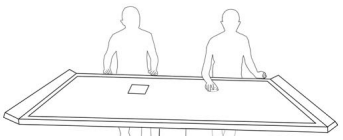
(e) Sender places tangible somewhere reciever can get it.



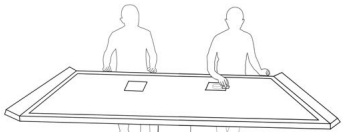
(f) Reciever can grab tangible whenever he wants.



(g) Reciever grabs tangible.



(h) Reciever places tangible on his workspace.



(i) Reciever presses paste button and can use the copy of the image on his workspace.

Figure 2.9: Deposit.

Chapter 3

Tangilow

Tangiflow is a tangible-based implementation of visual flow-based programming. It allows performing general-purpose programming via Python on a Microsoft Surface Hub. Our framework provides a possibility to work with different data types, such as basic Python data types and image files. Furthermore, the system supports the addition of other types of data and files in the future by developers, and the possibilities of it is bound only by possibilities of Python. A program in Tangiflow is built as a directed acyclic graph (Figure 3.1). Nodes of this graph represent functions. As data passes through nodes it is being processed in nodes. Then, data is passed to another node through edges that define the route of data in the graph.

3.1 Tools and Environment

Tangiflow uses several pieces of hardware. The system is running on iMac with OS X. Microsoft Surface Hub 84″ is used for displaying the user interface. We are using 4 passive tangibles. The iMac has Windows virtual machine installed on it, which is used to detect touches on Microsoft Surface Hub.

The software part of the system is based on Python and

The front-end is built on Swift, and the back-end is built on Python.

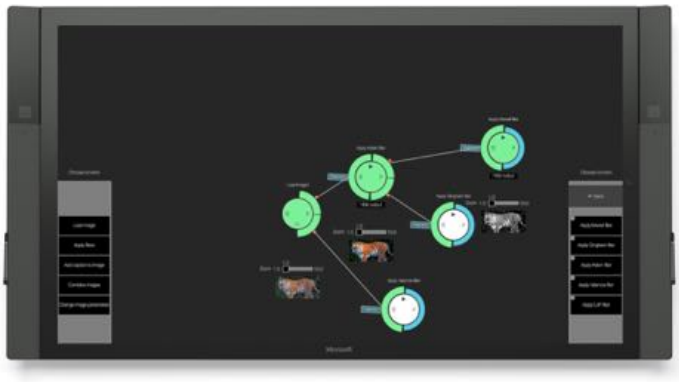


Figure 3.1: The workspace of Tangiflow.

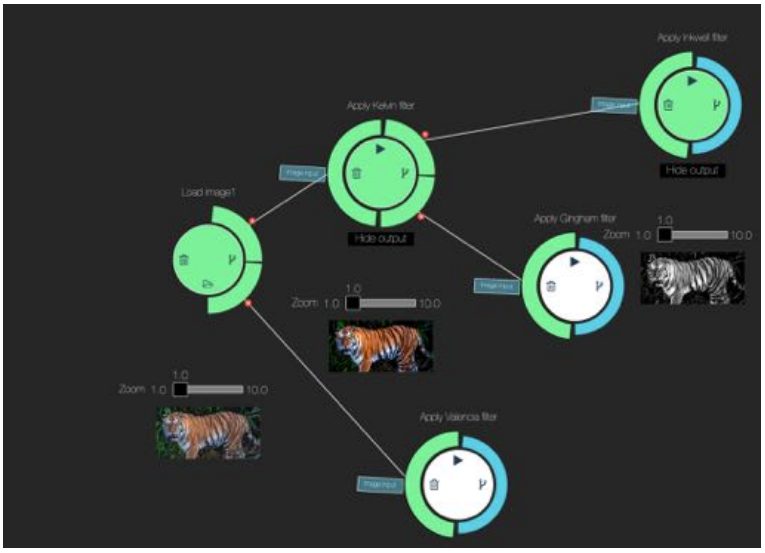


Figure 3.2: More detailed view of a simple program in Tangiflow.

Swift. The front-end is built on Swift, and the back-end is built on Python. Both of the languages communicate with each other using json files. Swift is a relatively young programming language, hence its collection of libraries is rather poor. Therefore, Swift was not suitable for the back-end of Tangiflow. Since Tangiflow was intended to be a general-purpose VPL, we needed a programming language that has a wide collection of libraries and frameworks for various purposes. This is why we picked Python to build the backend of our system.

3.2 SpriteKit

SpriteKit is used for the user interface.

Tangiflow's UI is based on Swift. Therefore, we are using SpriteKit[1], a framework developed by Apple for graphics rendering and animation infrastructure. It makes it easy to create high-performance, battery-efficient 2D games for IOS and macOS devices. We use this framework to draw and render the UI.

SpriteKit project consists of scenes. At each frame update, scenes pass through the rendering loop of SpriteKit.

SpriteKit project consists of scenes that are being initiated inside a view, which is an instance of the SKView class. SKView object handles all the rendering and animation process and refreshes the frames. Each frame goes through the rendering loop, where the content of each frame is being processed (Figure 3.3). At the end of each loop iteration, we get a SKScene object which contains all the objects of the scene and related data to draw them on the screen.

Objects on the scene in SpriteKit are called.

The main building blocks of SpriteKit apps are nodes, which are the instances of SKNode class. They can be combined into more complex nodes on the scene. We are using nodes to build UI elements such as nodes of graphs, edges of graphs, menus, labels, sliders, and etc.

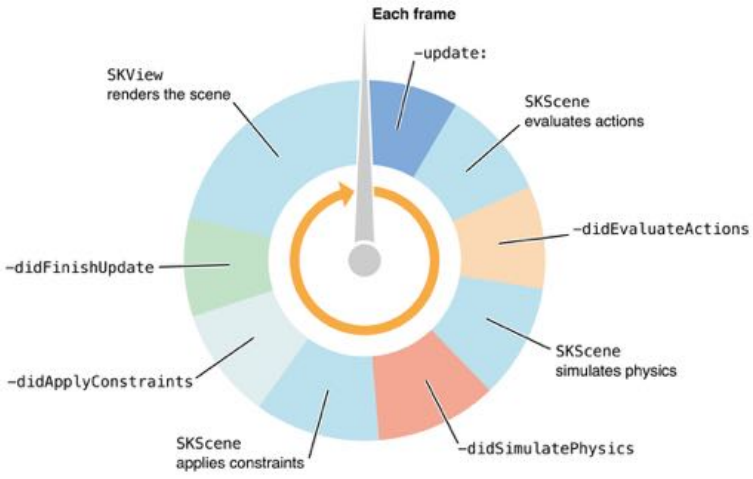


Figure 3.3: Processing loop of SpriteKit.

[1]https://developer.apple.com/documentation/spritekit/

3.3 MultiTouchKit

MultiTouchKit handles touch events and manages tangibles.

To detect and handle touch events from Microsoft Surface Hub and also to handle tangibles, we needed a framework. The framework used in our project is MultiTouchKit or abbreviated MTK. As the name implies, its main task is to receive and process touch inputs. MTK stores data about tangibles and touch events appearing on the screen. This framework also provides useful data, such as the position of touchpoints and tangibles, their ids, state, the time they have been created, etc.

Figure 3.4: MTK update loop scheme.

MTK is a wrapper around SpriteKit.

MTK was built as a wrapper around SpriteKit, thus it shares common ideas with the latter. As a result building apps on MTK does not differ much from building apps on SpriteKit. It uses the same concept of scenes but with a slight difference. In SpriteKit, scenes are instantiated from the SKScene class, but in MTK all scenes are instantiated from MTKScene, which is the modification of SKScene. MTKScene objects store and manage all spatial and logical relationships between all elements of the scene, which are

called nodes. The general structure of the framework can be seen in figure 3.4.

Previously we have mentioned that PUCs and PERCs tangibles are supported by MTK. In our project, for the sake of simplicity, we are only using PUCs. Our implementation of PUCs can be seen on figures 3.5 and 3.6.

Figure 3.5: PUCs which were developed for Tangiflow.

3.4 Spotify Pythonflow

Spotify's Pythonflow is used to build the back-end.

With the back-end, we decided not to reinvent the wheel and picked existing solutions for dataflow programming. Pythonflow is the simple implementation of dataflow programming by Spotify [2]. This framework provided us with all the functionality we needed to build a strong back-end for Tangiflow. One of the advantages of this framework is the *automatic caching* of computationally expensive operations, which can help to boost the performance of the framework while performing complex tasks.

Pythonflow programs are usually in the form of a directed acyclic graph.

In Pythonflow, building and running a dataflow program

[2] https://pythonflow.readthedocs.io/en/latest/

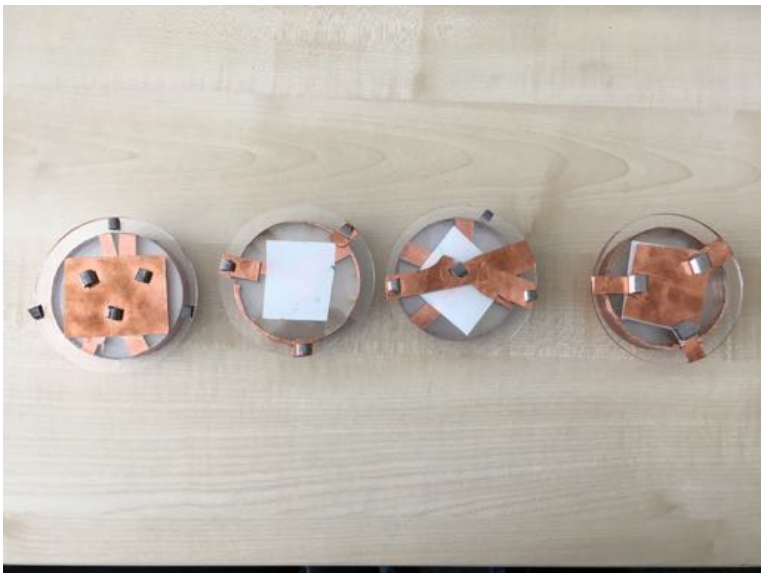


Figure 3.6: The bottom side of tangibles of Tangiflow. Pads and copper foil create a pattern that can be detected on a capacitive touchscreen.

usually is performed in two steps. Initially, we build a directed acyclic graph of operations that defines the logic of the program. Those operations are also called nodes. Nodes are connected between each other via edges, which create dependencies between operations. Following this, we provide input values for those nodes and perform computations.

3.5 Front-end

3.5.1 Workspace

Workspace of Tangiflow is very simple and consists of 2 toolbars.

While designing the UI of Tangiflow, we wanted it to be simple, so it would be easy to learn to use. The workspace of Tangiflow is pretty simple. It consists of two toolbars, which contain all the functionality available for work. The rest of the space on the screen is empty, and users can use it as a canvas to build their programs.

When we were designing Tangiflow, we decided to make the toolbar as a one-level menu. There was only one instance of the menu in the workspace. To find the needed item, users could navigate through the list of available items using navigation buttons. Since the menu was not organized in any way, it was hard to find the right tools among lots of items. It was also hard for two or more users to work since only one side of the tabletop had the menu.

The first prototype of Tangiflow had only one toolbar.

Figure 3.7: Initial version of workspace. There is one toolbar that stores all available functionality. The green button bellow toolbar is the run button.

In the early design of Tangiflow, we also had one global run button. After making changes to the program, users needed to press the run button to see the output of the program. In the case of multi-user usage of the system, it became an issue. To run their program, users standing far from the run button had to walk around the tabletop or ask the person standing near to press the button. In the later iterations of the design, we removed the global run button and added a run button to all nodes. Later we will talk about it more.

Early versions of Tangiflow had only one run button.

Since the early version of the UI had usability issues, we made some changes to our design. At the later iterations of the design, for the sake of convenience, we have grouped similar functions in the menu (Figures 3.8 and 3.9). As we mentioned before, we have also removed the global run button. Later in this chapter, we will explain how we run the program.

In the last version of Tangiflow we have solved existing problems.

The usage of Tangiflow does not differ from the usage of

The usage of Tangiflow does not differ from the usage of any regular VPL

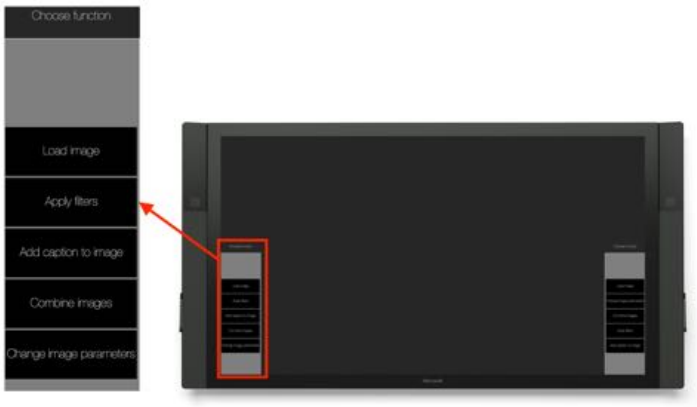


Figure 3.8: The final version of the workspace. There are two toolbars, and similar functions are grouped.

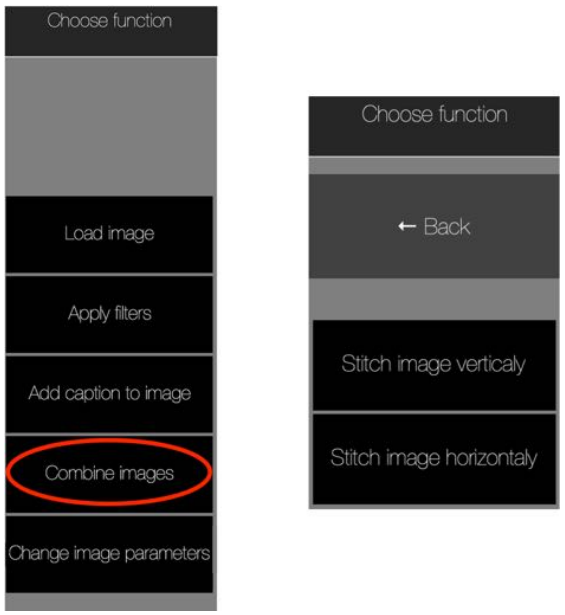


Figure 3.9: An example of two level menu in Tangiflow. After pressing "Combine images" button, menu with respective functions opens.

any regular VPL. Users select functions from the toolbar, and functions appear on the screen as nodes. Using those functions, users can pick data sources and files. Users can connect nodes between each other and disconnect them, creating a directed acyclic graph, which will process the

data. They can delete nodes, move them around. So Tangiflow has all basic interaction techniques a VPL would need.

3.5.2 Nodes

As we mentioned before, the program in Tangiflow is represented as a graph. Nodes are the building blocks of the graph. An example of a node can be seen in figures 3.10 and 3.11.

Tangiflow has 2 types of nodes: source nodes and processing nodes.

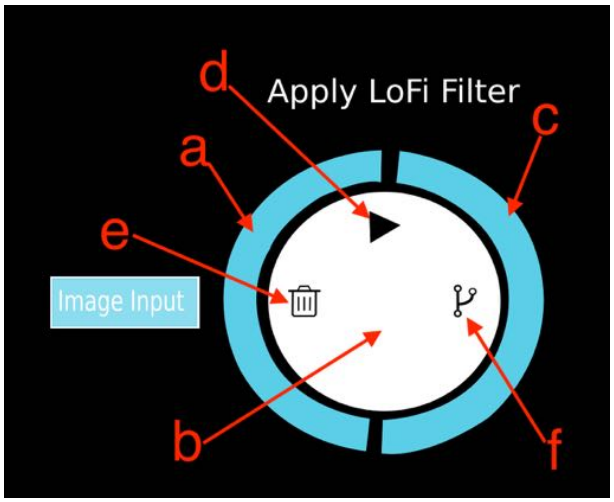


Figure 3.10: It is a sample of the processing node of Tangiflow. Nodes in Tangiflow are virtualy divided into three parts. (a) is an input arc, (b) is the body of the node, and (c) is the output arc. As it can be seen, node also has some buttons on it. (d) is the run button. (e) is delete button, and (f) is branch button.

Except for source nodes, other nodes are virtually divided into 3 parts: input arcs, body and output arcs. Input arcs are positioned to the left of the body and output arcs to the right, which can be seen on figure 3.10. This design decision was made due to the positioning of inputs and outputs in desktop VPLs we have surveyed. In most of them, the flow of the program goes from left to right. Therefore, input ports are placed on the left side and output ports on

Nodes consist of input arcs, output arcs and body.

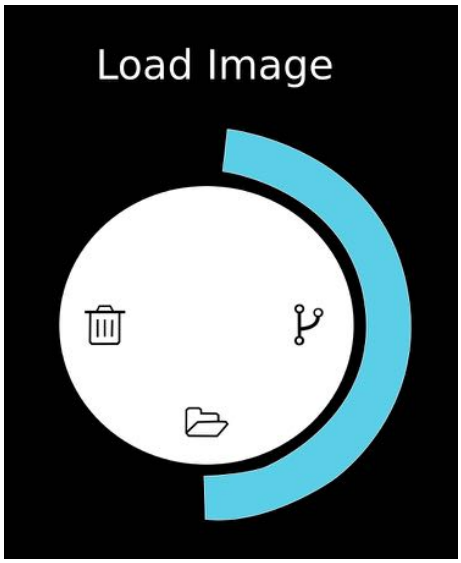


Figure 3.11: A sample of source node. As it can be noticed, source nodes don't have input arc. Furthermore, source nodes don't have run button, but they have a file manager button with folder icon on it. This button opens a file manager from where users can pick a file to work with.

the right side, which can be seen on figures 3.12, 3.13, 3.14. Nodes can have an arbitrary amount of input arcs and up to 6 output arcs. The body of the node can contain four types of buttons: delete, branch, run, open file manager.

Users can move nodes, connect them between each other and remove them.

Users can interact with nodes in several ways. Nodes can be moved inside workspace by placing the finger at the center of the body and dragging it. We have left the center of the body empty and positioned buttons closer to the borders to prevent accidental button clicks during movement of nodes. They can accept different types of input but always provide the same output no matter how many output arcs it has. Users can increase the amount of output from one node by pressing the branch button, and they can decrease the number of output arcs by pressing the close button, which appears when the node has more than one output arc (Figure 3.16). Output arcs of nodes can be connected to input arcs by dragging finger from one arc to another, and the connection can be broken with the same gesture. Arcs which are connected with edge expand and

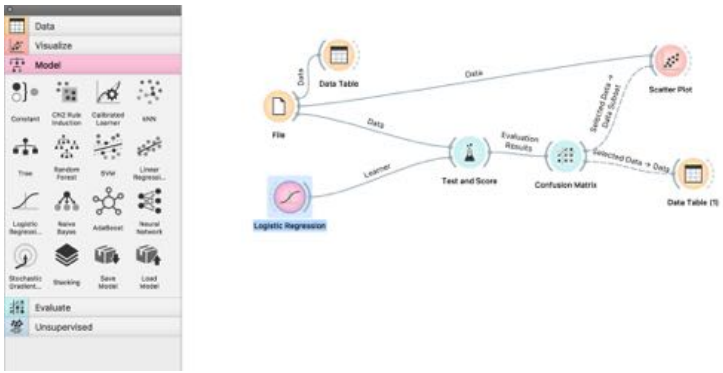


Figure 3.12: Sample of program in Orange.

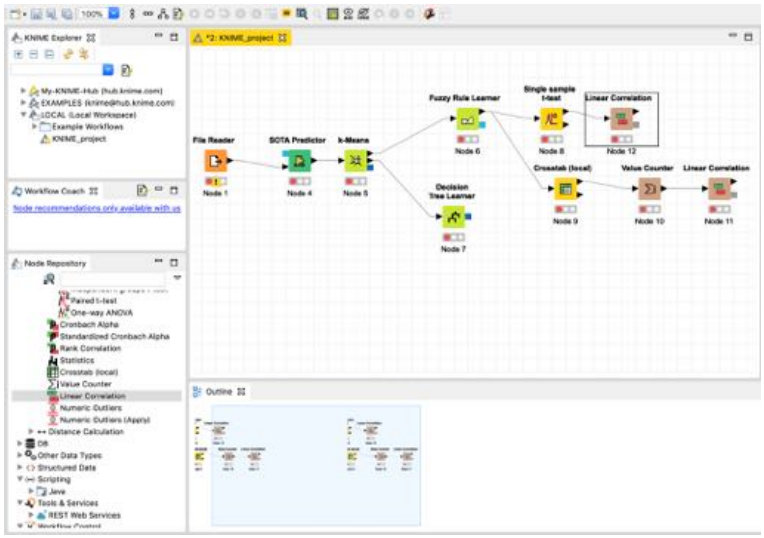


Figure 3.13: Sample of program in Knime.

Figure 3.14: Sample of program in RapidMiner.

change color (Figure 3.15)

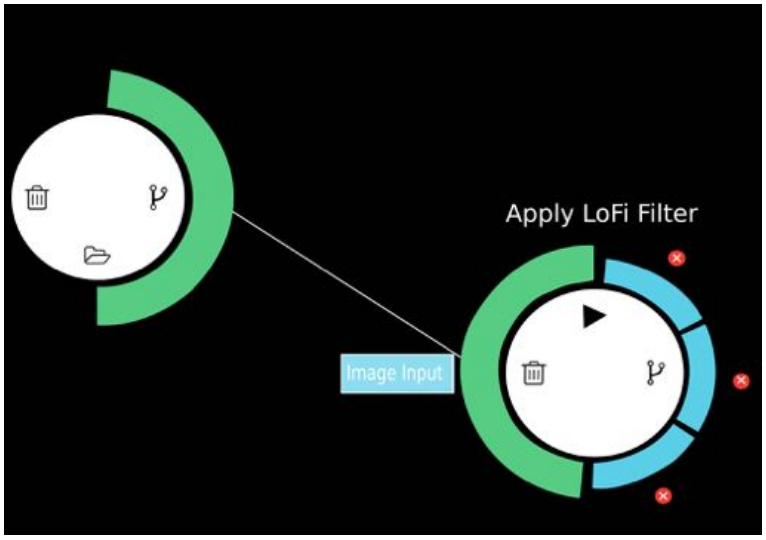


Figure 3.15: Two connected nodes. As can be seen, connected arcs are bigger and have a different color.

Tangiflow has two main types of nodes: **source nodes** (Figure 3.11) and **processing nodes** (Figure 3.10).

Source nodes are nodes which provide data.

Source nodes don't have input arcs because they play the role of data providers themselves. Since they do not do any action on data, source nodes do not have a run button on them. On the bottom part of the node's body, users can see the folder icon after pressing which file manager appears from where users can pick files to work with(Figure 3.17).

Processing nodes can not provide data by themselves. Instead, they get data from source nodes

As the name implies, processing nodes are processing data. They can not carry data by themselves, so they always have at least one input arc and don't have a file manager button. We have mentioned before that the global run button was removed. We have placed a run button on each processing node so users can run any branch of the graph by just pressing the run button of an individual node and it will not affect other branches of the graph.

Processing nodes can have input parameters.

Some of the processing nodes can have parameters that users can change. There are 3 different types of parametrized nodes. They differ with ways users can provide parameters. The first type of parametrized node is the

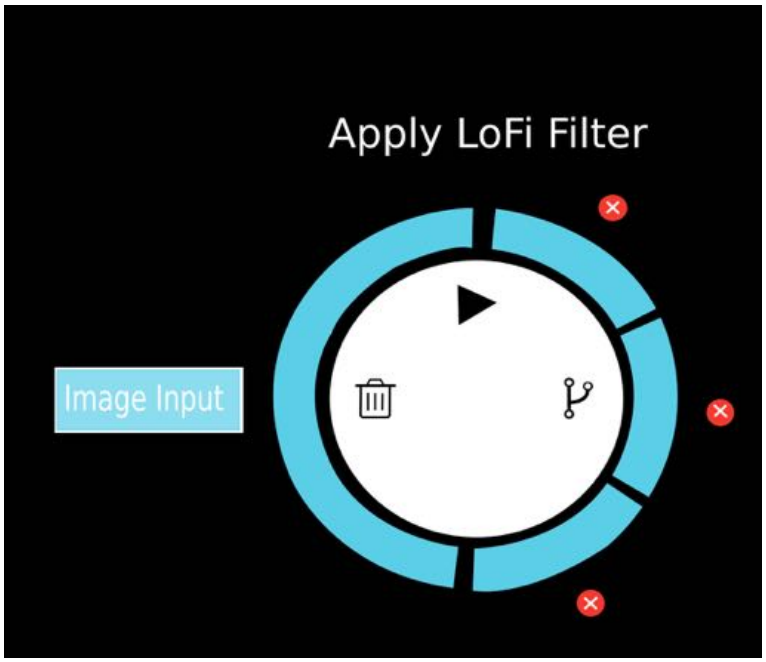


Figure 3.16: A sample of node after branching. It has more than one output arc and each arc has close button. Users can delete the branch pressing close button.

Figure 3.17: The file manager of the source node. After pressing the folder icon on the source node, the file manager opens. Available files are grouped in folders.

node with text field input (Figure 3.18). The second type of parametrized node is the node with slider (Figure 3.19). Those types of nodes can have an arbitrary amount of sliders. Range and step of sliders are defined by users. The last type of parametrized nodes is hybrid nodes. This type of nodes can accept parameters using both slider and text field (Figure 3.20).

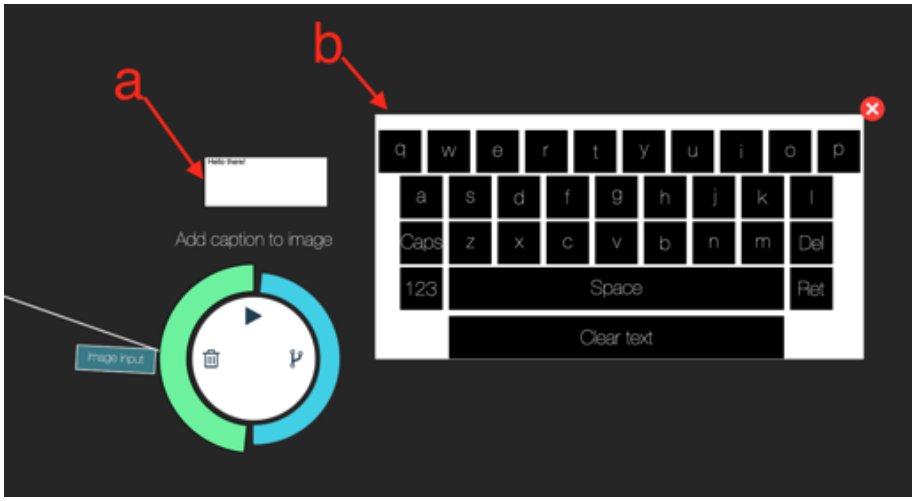


Figure 3.18: Node with text field and QWERTY keyboard. After tapping text field (a), keyboard (b) appears.

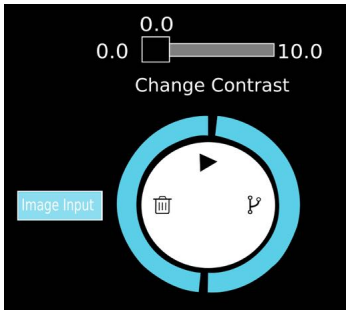


Figure 3.19: Node with slider input.

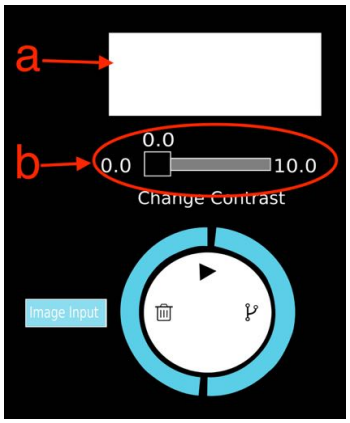


Figure 3.20: Hybrid processing node. (a) is a text field, and (b) is a slider.

3.5.3 Tangible Interaction

We use tangibles as data sharing medium in Tangiflow.

In Tangiflow tangibles are used as data sharing medium. Users, in the process of processing data, can upload data to tangible and pass them to other users. The procedure is very simple. The first user places tangible on an image he wants to copy after this copy button appears on the left side of tangible, which is needed to be pressed to copy data. Following this, the user places tangible on an empty space of the workspace, and then the paste button appears on the right side of the tangible, pressing which the user can paste the image. After pasting tangible creates the source node with the image copied to the tangible.

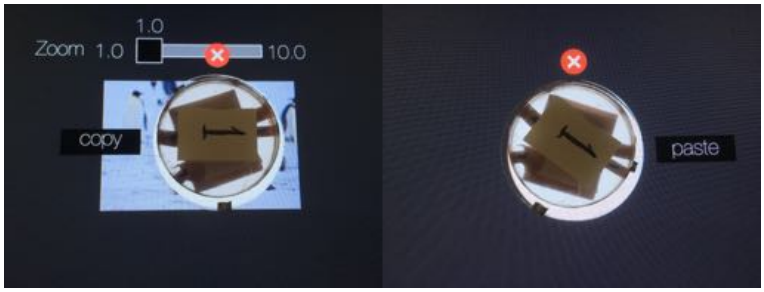


Figure 3.21: The usage of tangibles in Tangiflow. When a user places tangible on a photo, a copy button appears on the left side of the tangible. When the user presses this button, tangible stores this image. When tangible has an image copied on it, and the user places tangible on an empty place on the workspace, a paste button appears, pressing which the user can paste the image.

3.6 Back-end

The back-end of our project is written in Python. It is a *simple* language and has an *elegant* syntax. Nonetheless, it is quite *powerful* and has a *huge library base*, which can help to expand the functionality of Tangiflow in the future.

Tangiflow needs some files to function properly. The functionality of Tangiflow is stored in file *Subroutines.py* as Python subroutines. Each subroutine in *Subroutines.py*

matches to each available node in Tangiflow. When a user wants to run some function, the back-end calls it from *Subroutines.py*.

Front-end and back-end communicate using JSON files.

Since we are using two different programming languages for back-end and front-end, they need to communicate somehow. For this we are using JSON files. The first JSON file is *myproj.json*. This file stores information about available functions for users in Tangiflow. The front-end uses this file to build UI. The back-end uses this file to match nodes to subroutines in *Subroutines.py*.

Programs in Tangiflow are stored as graph structure in JSON files.

The second file is called *graph.json*. When the user runs program in Tangiflow, the front-end collects data about the graph structure user has built and stores it in *graph.json* file. The created file also stores information about parameters and arguments each function accepts. The back-end uses this file to build a graph structure which can be read by Pythonflow.

Both JSON files are essential for Tangiflow to function. After receiving those files,the back-end decomposes the graph into branches and then decides which one to run based on data obtained from JSON files. After execution, results are stored in JSON files that are matched with nodes to which they belong. Front-end collects those files and provides users with output, an example of which can be seen in figure 3.1.

Users can run branches of programs independently.

While designing Tangiflow, we wanted to give users possibility to run branches of the program independently. Let's say we have built a program that can be represented as a graph in figure 3.22. If the user presses run button on the node D, it will not affect other branches, and only the branch ABD will run. Later, if the user decides to press the run button on the node B. it will not affect the branch ABD entirely but will only run AB. It was done due to performance reasons, and it reduces the execution time of the program. More details about how Tangiflow is functioning can be found in the documentation.

TangifLow is open for functionality expansion.

We wanted Tangiflow to be flexible. This is why Tangiflow can work with a variety of data structures, data types, and

files. At the moment. it supports generic Python data types, existing Python data structures and custom data structures created by users. It can also process image files and UI fully supports the output of image files and generic Python data types. Support of audio, video and other file types can be added in the future if desired.

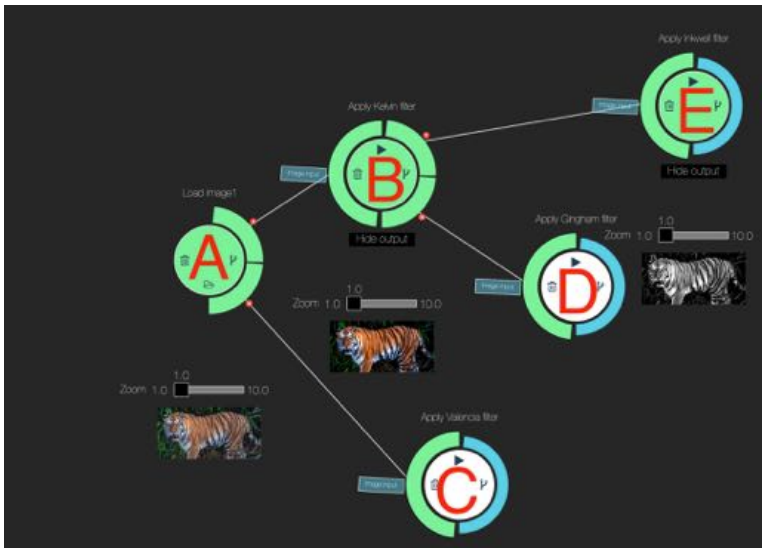


Figure 3.22: A sample of program in Tangiflow. This program has three branches: ABE, ABD, AC. User can execute any branch of this program.

Chapter 4

Evaluation

In the previous chapter, we discussed TangiFlow. This section will inform readers about the user study that we conducted to elicit how a group of 2 or 4 people shares data using tangibles. We also introduce new interaction techniques for tangible data sharing we have discovered during user study and try to explain the reason behind performing those interaction techniques. The secondary objective of the study was to see how people perform with the system in general, and what are their thoughts about TangiFlow.

4.1 Research Questions

We were interested in how users share data when they have more than one tangible. So we decided to shed light on the questions like: Does the growing number of users lead to an increase in the number of tangible usages? If yes, does it make sense to increase the number of tangibles? If we increase the number of tangibles, will it lead to confusion among users?

4.2 User Study

4.2.1 Setup

Our setup consists of horizontally positioned Microsoft Surface Hub 84". It has a screen of size 220x117 cm with a resolution of 3840x2160 pixels. This tabletop can detect up to 100 touchpoints. We did not use the computing power of Surface Hub, but instead, we were using iMac Pro to run the software. For data sharing tasks, 4 tangibles were used. The whole study process was recorded on a digital camera.

Figure 4.1: Two-user user study.

4.2.2 Participants

In total, 18 participants aged from 18-25, 13 males and 5 females took part in the user study. 7 of them had previous experience in image editing, 4 of them had at least some experience with VPLs, and 5 of them have worked with TUIs before. We had 6 groups of 2 participants and 3 groups of 4 participants. Overall, 9 studies were conducted and in 6 of them, participants have met each other before. 6 participants from studies with groups of 2 participants were also

Figure 4.2: Four-user user study.

involved in the studies with groups of 4. Studies with each group consisted of 3 parts. In the first part, they were provided with 1 tangible, in the second part with 2 tangibles and in the last part with 4 tangibles. Later in our thesis, we will refer to our participants with unique ids, an example of which can be seen in figure 4.3.

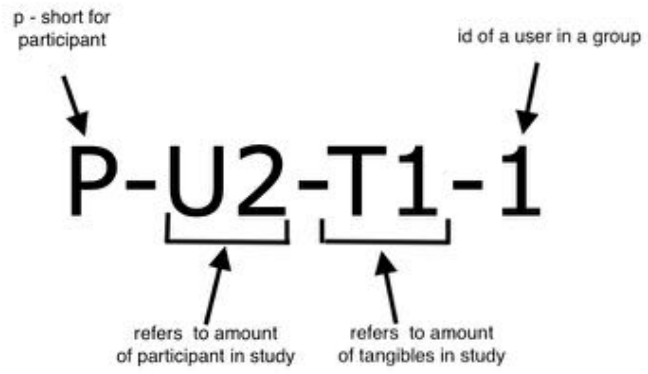


Figure 4.3: Example of unique ids of participant.

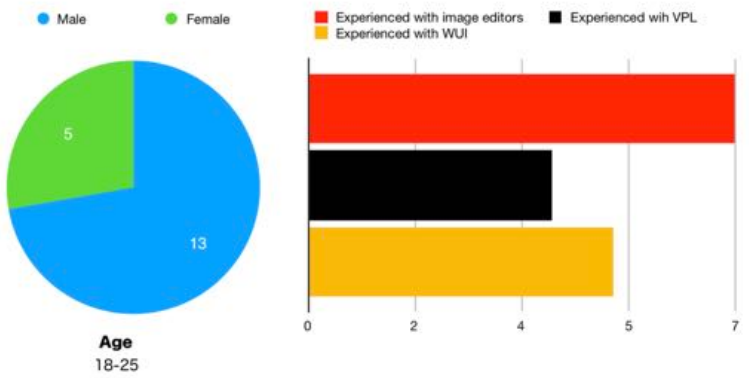


Figure 4.4: Demographic information of the user study.

4.2.3 Procedure

Inform users about our research: Users were informed about the purpose of the study, and they were also briefly explained what are the expectations from them at the user study. Following this, they were asked to fill in a form with demographic questions (Appendix A).

Train users: To establish a basic knowledge about TangiFlow, we have taught functionalities of our system to participants, and to reinforce obtained knowledge we asked them to perform small tasks like applying a filter to the image, adding a small caption to it.

Ask them to perform tasks: After users became familiar with TangiFlow, they proceeded to the main tasks. Users needed to accomplish a series of creative and collaborative image editing tasks. They were also explicitly asked to perform any data exchange with tangibles.

Collect feedback: If participants were exchanging data in an interesting or unexpected way, they were asked what their thought process was. Also, information about how many tangibles was more convenient for them to use was obtained from participants.

4.2.4 Study Tasks

Each study session involved image editing tasks. Participants were asked to create a small comic (4-6 images) from an existing set of images. Each participant needed to pick 2-3 images and then edit them in the desired way, following it their task was to copy the edited image using tangible and share it with another participant. The task of receivers was to add captions to the received image. In the end, they were required to combine images in one page of a comic. In the first part of the study, we provided users with one tangible, in the second with two tangibles and in the last part with four tangibles. At each part of the study, they were provided with a different set of images. A sample result of those tasks can be seen in figure 4.5.

Objective of our study was to create comics consisting of 4-6 images.

During the study, participants were observed by the principal investigator. Interesting or unusual behavior of participants were noted down. Furthermore, at the end of each session participants were asked questions about those behaviors. The whole process of the study was recorded on a camera. Later all videos were examined manually by the principal investigator. Each video was examined several times and interesting moments were recorded for further investigation. Following this, we started to search for the appearance of moments that interested us and interesting behavior in other videos. We were interested if other participants acted the same way in other study sessions too, were there any similarities between the actions of participants.

4.3 Results

4.3.1 Users' Feedback

TangiFlow got mostly favorable feedbacks from users. Participants liked the general idea of the framework and found most of the design decisions and interaction techniques intuitive. Among the advantages of the system, they have

TangiFlow got mostly favorable feedbacks.

Figure 4.5: Result image from one of the studies.

highlighted responsiveness of the interface, color selection, easiness of building programs and immediate output of the result. They did not have issues while learning to work with TangiFlow. Furthermore, users appraised the possibility of copying and pasting intermediate or final results of the program and deleting the whole chain of actions, which lead to those results thereby saving the workspace. Some of the positive feedbacks from participants:

- ”I enjoyed working with your app. I liked the idea of putting an image editing program on a big touch screen.” -P-U2-T2-3
- ”It is cool that, when I press the run button my edited image shows up under the node immediately” - P-U4-T1-1
- ”When my part of the screen was full, I just copied the final image and pasted it. Then I just deleted all other nodes. It was really handy.” -P-U4-T1-5

Size of nodes was an issue for most of the users.

Along with advantages users also noted some drawbacks of our framework. One of them was the size of the nodes. They have found their size quite big. After building big flows, the workspace was overflowed with lots of nodes, and it was hard to keep track of them (Figure 4.6). Some of the negative feedbacks from participants:

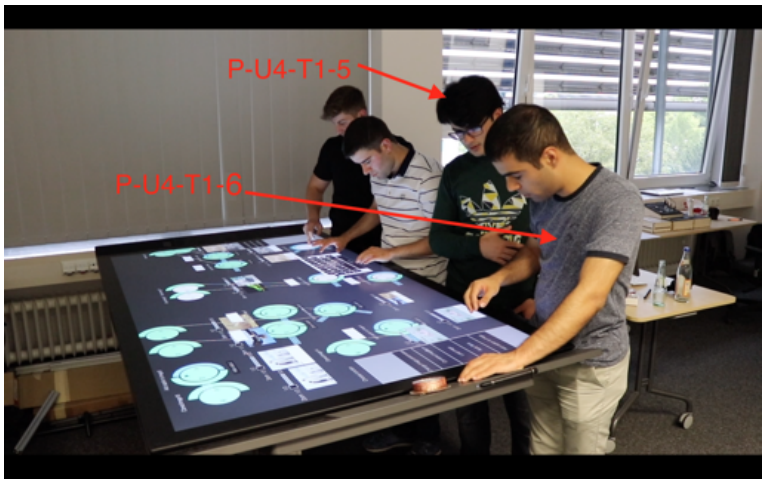
- ”Why do you need such big nodes? They are using so much space.” -P-U2-T1-2

- ”I think smaller nodes would be better. These nodes look big, and they fill the screen to fast.” - P-U4-T1-5

There were not enough toolbars for participants.

Another problem with the UI was the number of toolbars. We have 2 toolbars, and when 4 users work with TangiFlow, it is hard for users who are standing far from the toolbar to reach out for it. Furthermore, when there are 4 users and 2 of them are using toolbars, the other 2 have to wait (Figure 4.7). All users have found the keyboard developed for TangiFlow quite big and noted slow response time on the text field while typing on the virtual keyboard.

Figure 4.6: Workspace is overflowed with nodes which makes it harder to work.



(a)

(b)

Figure 4.7: In image **a** participant P-U4-T1-6 is strugling to choose function from toolbar. Participant P-U4-T1-5 is waiting for P-U4-T1-6 to make a choice so he can also use toolbar.In image **b** participant P-U4-T1-6 has already made his choice and participant P-U4-T1-5 can use the toolbar.

4.3.2 Observations

As we mentioned before, our main interest in this research was the behavior of users depending on the number of tangibles. However, during studies, we have come up with

some unexpected findings which we would like to address first. First, we will try to answer the question: Are users willing to share their tangibles? Following this, we will address our main research question.

Figure 4.8: Handoff process during study.

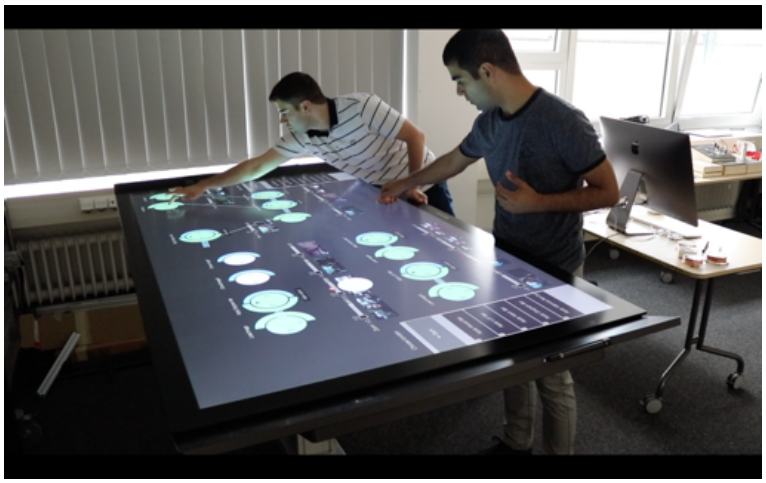


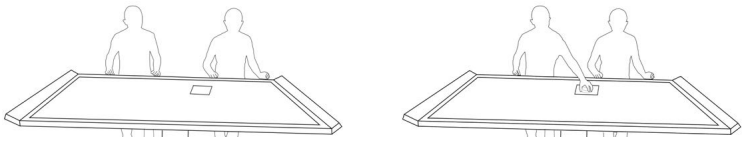
Figure 4.9: Deposit process during study.

Are users willing to share their tangibles ?

To answer this question, let us first introduce you to new techniques we have observed during user study. As was discussed in previous chapters, we were expecting two data sharing techniques to be performed by participants:

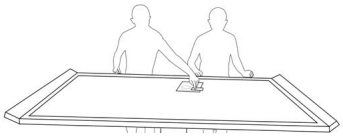
We have found 2 new interaction techniques: abduction and forced deposit.

handoff and deposit. However, it was not always the case. We have found two more types of data sharing techniques. The first one we called **abduction** (Figure 4.10) and the second one **forced deposit** (Figure 4.11). Abduction is the interaction when the first person invades the second person's area, copying data the second user worked on and then pasting the data at his workspace. Forced deposit is the interaction when the first person copies data from his side of tabletop and pastes it in the second person's working area. Occasionally users were not willing to give away their tangibles and were performing abduction and forced deposit instead of handoff and deposit.

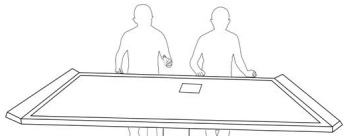


(a) Sender has an image but he is busy to send the image.

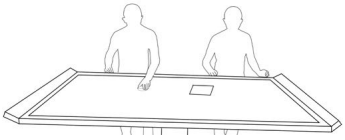
(b) Reciever gets the tangible .



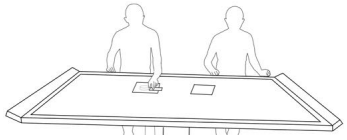
(c) Reciever copies photo.



(d) Reciever brings tangible from senders side of workspace to his.

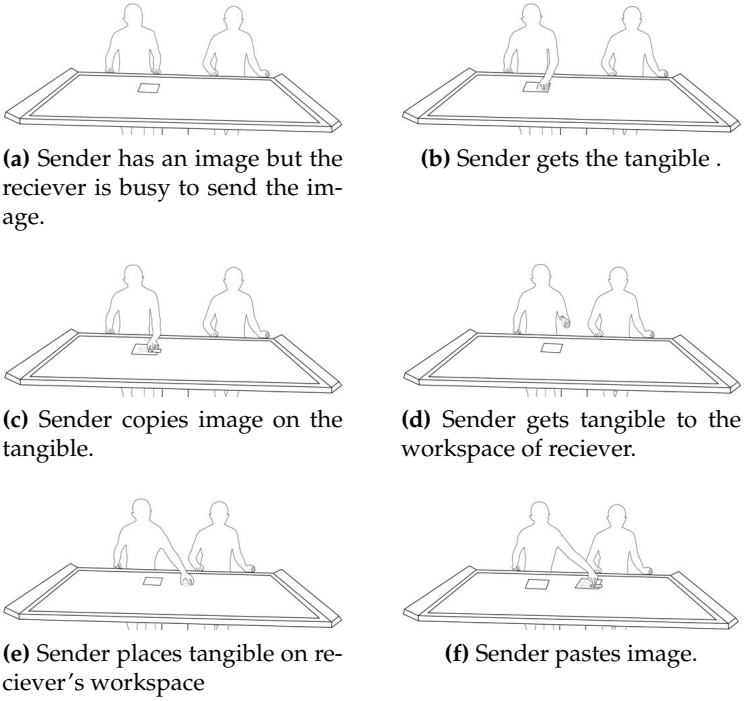


(e) Reciever places tangible on his workspace



(f) Reciever pastes image.

Figure 4.10: Abduction.



(a) Sender has an image but the reciever is busy to send the image.

(b) Sender gets the tangible .

(c) Sender copies image on the tangible.

(d) Sender gets tangible to the workspace of reciever.

(e) Sender places tangible on reciever's workspace

(f) Sender pastes image.

Figure 4.11: Forced deposit.

The overall amount of each interaction type can be seen on figures 4.12 and 4.13. Of course, the number of abduction and forced deposit interactions may seem insignificant in comparison with the number of handoffs and deposits, nonetheless, they have occurred frequently enough to notice them and start to dig the reasons behind those interaction techniques.

	1 Tangible	2 Tangibles	4 Tangibles	Overall
Deposit	15	4	23	42
Handoff	18	30	16	64
Abduction	7	0	4	11
Force deposit	1	2	2	5
Total				122

Figure 4.12: The amount of different tangible interactions in user study with 2 participants. Overall 6 studies were performed.

	1 Tangible	2 Tangibles	4 Tangibles	Overall
Deposit	2	3	3	8
Handoff	11	12	14	37
Abduction	0	0	1	1
Force deposit	8	7	12	27
Total				73

Figure 4.13: The amount of different tangible interactions in user study with 4 participants. Overall 3 studies were performed.

Handoff and deposit are interaction techniques when users give away tangible. Abduction and forced deposit, however, are the interaction techniques where the user does not give away tangible. For the sake of convenience, we will call those groups **giveaway** and **non-giveaway** interaction techniques.

We grouped interaction techniques as a giveaway and non-giveaway interaction techniques.

If we examine closely the numbers we can notice that the total amount of giveaway interaction techniques decreased dramatically in studies with groups of 4. Furthermore, the total amount of non-giveaway interaction techniques increased. We can say that increasing the number of users affected their willingness to share tangibles.

In studies with groups of 4, the amount of abduction decreased dramatically, and the amount of forced deposit increased.

If we compare non-giveaway interaction techniques we can notice that abduction appeared more in studies with groups of 2, and in studies with groups of 4, participants performed abduction only once. In the studies with groups of 2, it is easier for the receiver to see if another person has finished his work with an image. On the contrary, in studies with groups of 4, it was harder to keep track if someone finished working on image and if data is ready for abduction. On the other hand, the amount of forced deposit increased sharply. In contrast to abduction, in forced deposit the sender always knows if the image he is working on is ready to be deposited or not, so participants tend to perform forced deposit instead of abduction in studies with groups of 4.

In the abduction process, participants need assurance that the sender has finished the work on image.

Let's examine figure 4.14. This is the study where the abduction interaction technique appeared the most. Partici-

pant P-U2-T2-2 finished working on 3 images and grouped them. Nodes with those images outlined with a green frame. After participant P-U2-T2-2 was done with images, he informed participant P-U2-T2-1 that she can use images, and she started to abduct images from participant P-U2-T2-2. It shows that the abduction process is appearing when both parts are sure that a piece of data sender was working on is ready for transfer.

Figure 4.14: Abduction process during study.

If we look at figure 4.15 we can see that P-U4-T2-2 is delivering the image to P-U4-T2-3 while P-U4-T2-3 is involved in some discussion with P-U4-T2-4 and in contrary to abduction he does not need to be informed by P-U4-T2-2 that data is ready for usage and P-U4-T2-2 can freely deliver the image.

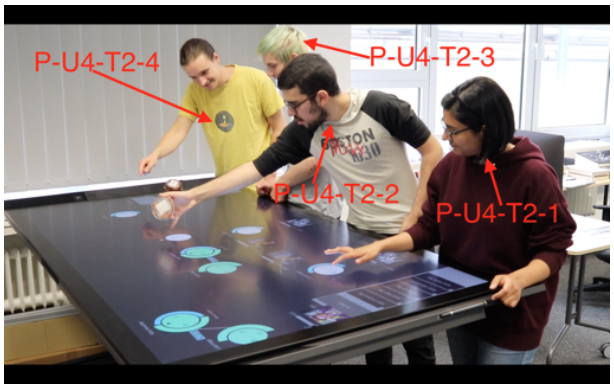


Figure 4.15: Forced deposit process during study.

Based on the feedback and comments from participants, we have established two main reasons why people performed abduction and forced deposit:

When each user has his own tangible, they can build ownership feeling.

1. **Ownership feeling.** Performing non-giveaway interaction techniques is the manifestation of ownership feeling. Lack of tangibles made users share since they did not have any other means to exchange images with other users, but as soon as each user got their own tangible, they started to build an ownership relation with it, therefore, they did not want to give it away. Participants were afraid that their tangible will mix up with other tangibles .

   One of the solutions for this problem could be creating tangibles with different shapes or colors, so it becomes easier to distinguish them, and users may find their tangible way easier than before. However, according to users, different colors and shapes could have led them to the thought that different tangibles with different shapes and colors may serve different purposes. Another solution for this problem is to provide users with fewer tangibles than the number of users, but not as few as not to create a bottleneck during the interaction. For example, in the study with 4 participants, 2 tangibles were enough to eliminate this problem and it did not create a bottleneck.

2. **Busyness of other users.** When the user to whom other users wanted to pass data or from whom he wanted to get data was busy, then he was just performing the delivery of the data himself without disturbing that person. We could not come up with a solution to this problem, and this problem can be addressed in the future by other researchers.

Does the growing number of users leads to an increase in the number of tangible usages?

Increase in the number of users did not lead to an increase in the amount of tangible usage.

An increase in the number of users, in general, did not lead to an increase in the amount of tangible usage. In the case of the user study with groups of 2, the average amount of tangible interaction was approximately 20 per

study and in the user study with 4 participants, this number was around 24 per study. So as can be seen, the difference between numbers is not dramatic. Overall, if we look at table 4.12 and 4.13, we can see that when the number of tangibles matches the number of participants they perform handoff interaction more frequently than in other cases. Also, in the case of the study with 4 participants, forced deposit was the second preferable data exchange type.

Figure 4.16: Another bottleneck situation where 1 participant is using tangible and other participant has to wait .

Two tangibles were enough for groups of two and groups of four.

The increasing number of users and lack of tangibles caused a bottleneck. In the study with groups of 4 and 1 tangible, when 1 participant wanted to share his data with another user, other users had to wait for that user to finish his part of the job (Figure 4.17). In the study with 2 participants and 1 tangible, a similar situation appeared but it was not as critical as in the study with 4 participants since fewer people were waiting for the tangible (Figure 4.16). The obvious solution for us to this problem was to provide each user with a tangible , but as it occurred we were wrong. This solution worked for the study with groups of 2. All users were equally involved in the case of study with 2 participants with 2 tangibles(Figure 4.18). However, it did not work out in all cases for 4 participants. When users needed to exchange data simultaneously, all of them were confused and no one knew which tangible carries which data (Figure

4.19). Surprisingly, in the user study with groups of 4, participants reported that they felt more comfortable working with two tangibles since it was easier to follow and coordinate fewer items.

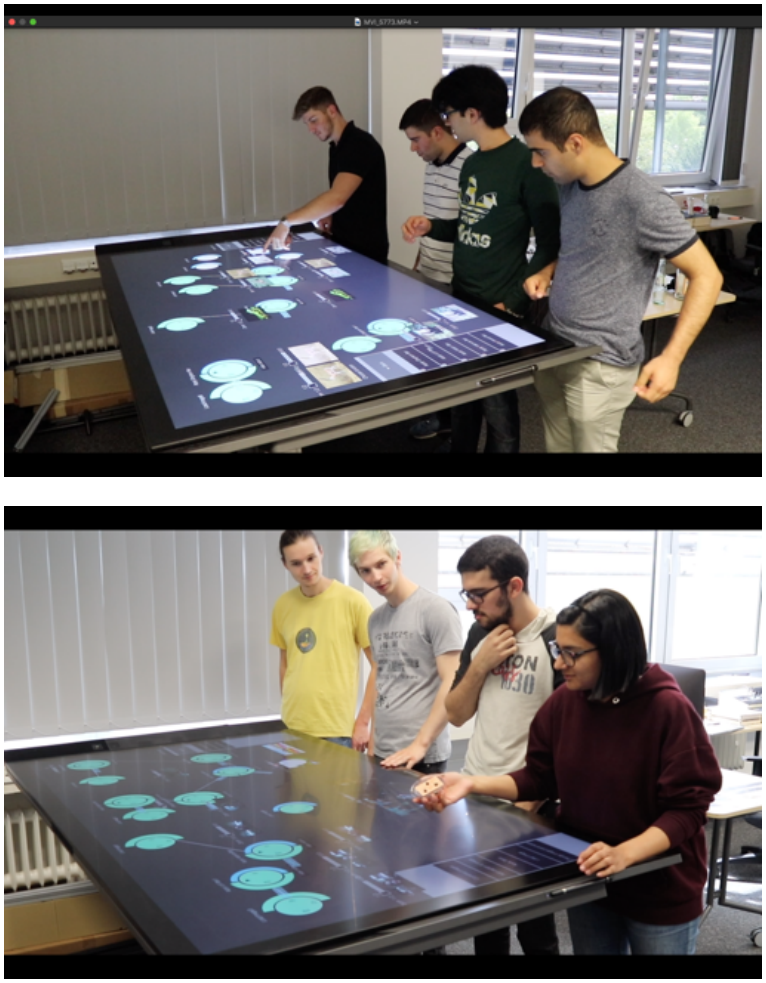


Figure 4.17: On this photos we can observe bottleneck situation. 1 user is using tangible and other 3 are waiting for their turn.

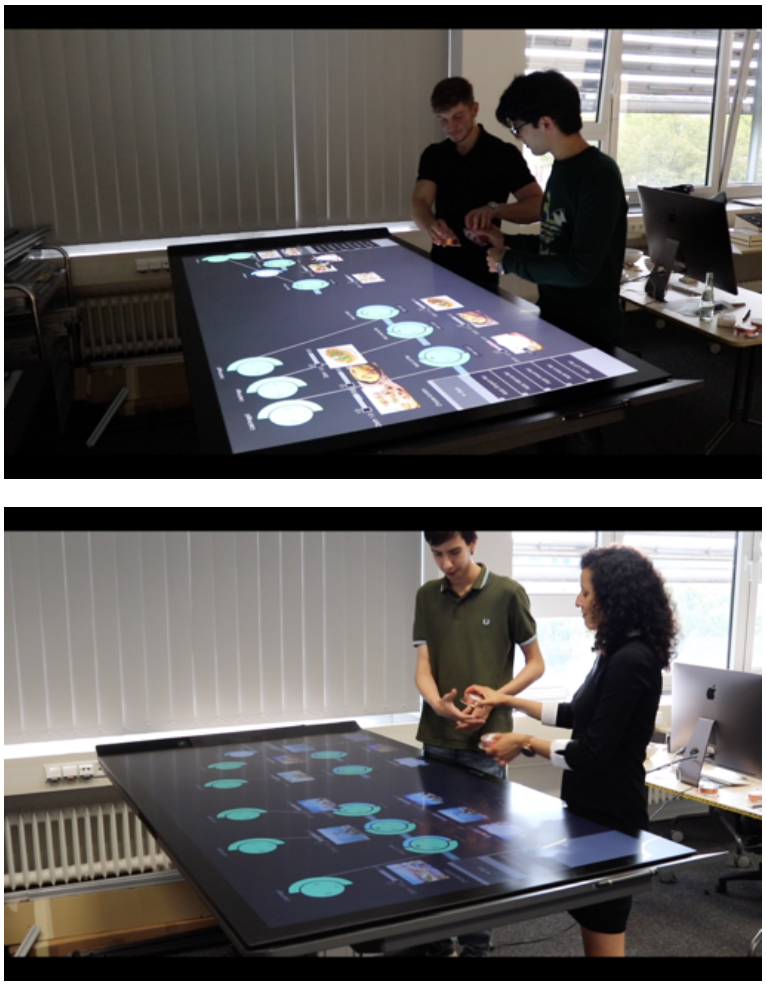


Figure 4.18: In the user study with 2 participants and 2 tangibles, users did not encounter bottleneck problem. On these photos 2 users are synchronously exchanging tangibles. Since each user has his own tangible and the amount of tangibles is not to much it is easier and faster to perform exchange of data.

Figure 4.19: Participants are confused and don't know who owns which tangible and to whom they should pass the tangible.

In studies with groups of four sometimes users were grouping in smaller groups of two.

In some cases, participants were gathering in groups of two, and in those cases, the process of tangible exchange with 4 tangibles went more comfortable without pauses and confusion among participants. The reason for that was that each group was using 2 tangibles and they were sharing data inside their small group so they did not have to keep track of the other 2 tangibles and after this, they were exchanging images between groups.

Overall, in studies with groups of two, participants did not like having extra two tangibles.

In the user study with 2 participants, we have also studied 4 tangible cases. In some cases, users ignored extra tangible. There were also cases when either during study or after it participants reported that they don't see the point of using extra tangibles however they kept using them. It was mostly because it was part of the task to use 4 tangibles to exchange data and users felt obliged to use all available tangibles.

4.3.3 Limitations of the Study

Our study was limited to a maximum of 4 tangibles and 4 users.

One of the main limitations of our study is the limited amount of tangibles. Limitations of the number of tangibles are due to the limitation of PUC tangibles. We have

addressed this problem in chapter 3 section of this thesis.

Another limitation of the study was the size of the tabletop . The screen size of the tabletop is not big enough to accommodate more users, furthermore, more people mean more nodes on the screen, which overcrowds the workspace with nodes and makes it hard to work. Users start to group nodes at the top of the screen. (Figure 4.20) These are the main reasons why we were studying 2 and 4 user cases.

Figure 4.20

Chapter 5

Summary and Future Work

The last chapter recaps the thesis and also gives ideas for future work.

5.1 Summary and Contributions

As we mentioned before, tabletops and TUIs are being used in more and more areas. We also have VPLs, which provide an environment for easy programming for novice users and helps to enhance their work for expert users. In this thesis, we tried to combine TUIs and VPLs and create a universal tool to work with different data.

In the first chapter, we discussed why we are interested in studying users' behavior while sharing data in a collaborative environment with tangibles. We also explained the motivation behind creating TangiFlow. TangiFlow provides general-purpose dataflow VPL functionality on a tangible user interface.

In the second chapter, we introduced some research done in the area of tangible based programming and discussed the limitation of existing solutions. We have also briefly ex-

plained the work of Subramanian et al. [2007] about tangible data sharing and noted points they did not investigate in their research and which we tried to explore more. This chapter also provides information about the survey of existing VPLs we have conducted and shows similarities between existing desktop-based VPLs , which we could use in TangiFlow .

The third chapter of this thesis described the working principle of TangiFlow . Here we talked about hardware, software, frameworks we used in the development of our system. Furthermore, we gave detailed information about the front-end and general idea of how the back-end works. In addition to that, we also explained some limitations of tangibles our system uses.

In the Evaluation chapter, we described the research questions that we wanted to answer. We then discussed the structure of our user study, its limitations. We found two new interaction techniques: abduction and forced deposit . Following that, we compared the frequency of those techniques to handoffs and deposits . Overall, handoffs and deposits were used dramatically more than new techniques. We also discovered the reasons for users performing those interaction techniques. We found that users perform abduction and forced deposit first of all because they build ownership feeling to tangibles, and secondly, they do not want to disturb other users while they are busy.

5.2 Future Work

In this section, we will briefly discuss possible future work with TangiFlow indexTangiFlow, additional features that can be added to enhance TangiFlow . We will also briefly look at possible future studies and problems other researchers might want to address in the future.

5.3 Implementation

For the sake of simplicity, we have only implemented functionality for image editing and generic python data type processing. TangiFlow has the potential to work with audio, video and statistical data. In the future, support for those file types can be added.

Tangible in TangiFlow copies and pastes only the result node of a flow. A good addition to the system would be the possibility of copying and pasting all of the flow which led to the final result. According to users' feedback, they also would like to have a possibility to hide intermediate nodes of the graph when the screen gets overcrowded.

The file manager of TangiFlow indexTangiFlow only supports one level folders, and the improvement of file manager to support arbitrary deep level of folders would be a good addition to our system.

5.4 User Study

We were able to conduct a study with a maximum of 4 users and 4 tangibles. It was mainly due to the size of the tabletop and the limitation of MTK . In the future, additional user studies with bigger tabletop , with more enhanced MTK could be done using more tangibles and involving more participants.

Also, in our studies, we used image editing tasks since TangiFlow only supports image editing and processing of generic python data types. We think with tasks like statistical data analysis, future researchers can get more insights about users' behavior while sharing data with tangibles. We believe that statistical data analysis and processing tasks involve more cognitive activity, which could lead to more interaction between participants. As a result, this could lead to more tangible data sharing interactions among users.

Appendix A

Informed Consent Form

Informed Consent Form

PRINCIPAL INVESTIGATOR Asif Mayilli
Media Computing Group
RWTH Aachen University
Email: asif.mayilli@rwth-aachen.de

Purpose of the study: Tangibles are physical devices which can represent digital data. The goal of this study is to find out how people use tangibles around tabletop and how they share digital data using tangibles. The results of this study will be useful for improving the user experience of tangible user interfaces.

Procedure: Participation in this study will involve three tasks. During all tasks you will be creating comics with another participant. This study should take about 50-60 minutes to complete. The session will be recorded on camera and the voices of participants will also be recorded.

Risks/Discomfort: You may become fatigued during the course of your participation in the study. You will be given several opportunities to rest, and additional breaks are also possible. There are no other risks associated with participation in the study. Should completion of the task become distressing to you, it will be terminated immediately.

Alternatives to Participation: Participation in this study is voluntary. You are free to withdraw or discontinue the participation.

Cost and Compensation: Participation in this study will involve no cost to you. There will be snacks and drinks for you during and after the participation.

Confidentiality: All information collected during the study period will be kept strictly confidential. You will be identified through identification numbers. No publications or reports from this project will include identifying information on any participant. Video and audio materials will solely be used for research purpose and will not be shared publicly anywhere. If you agree to join this study, please sign your name below.

_____ I have read and understood the information on this form.
_____ I have had the information on this form explained to me.

Participant's Name	Participant's Signature	Date
	Principal Investigator	Date

If you have any questions regarding this study, please contact Asif Mayilli at email: asif.mayilli@rwth-aachen.de

Survey

How old are you?

A) 18-24 years old
B) 25-34 years old
C) 35-44 years old

What is your gender?

A) Female
B) Male
C) Prefer not to say
D) Other: _____

Do you have technical background?

A) Yes
B) No

Do you have background in IT?

A) Yes
B) No

Do you have experience in image editing?

A) Yes
B) No

Do you have experience with Visual Programming?

A) Yes
B) No

Have you met other participant of the study before?

A) Yes
B) No

Do you have experience with Tangible User Interfaces?

A) Yes
B) No

Bibliography

Alistair D. N. Edwards. Visual programming languages: The next generation. *SIGPLAN Not.*, 23(4):43–50, April 1988. ISSN 0362-1340. doi: 10.1145/44326.44330. URL http://doi.acm.org/10.1145/44326.44330.

Sergi Jordà, Martin Kaltenbrunner, Günter Geiger, and Marcos Alonso. The reactable: A tangible tabletop musical instrument and collaborative workbench. In *ACM SIGGRAPH 2006 Sketches*, SIGGRAPH '06, New York, NY, USA, 2006. ACM. ISBN 1-59593-364-6. doi: 10.1145/1179849.1179963. URL http://doi.acm.org/10.1145/1179849.1179963.

Rene Linden. Multitouchkit: A software framework for touch input and tangibles on tabletops and. Master's thesis, RWTH Aachen University, Aachen, September 2015.

Paul Lyons, Giovanni Moretti, and Chrissy Reeves. Some possibilities of visual programming languages. In *Proceedings of the Symposium on Computer Human Interaction*, CHINZ '01, pages 43–47, New York, NY, USA, 2001. ACM. ISBN 0-473-07559-8. doi: 10.1145/2331812.2331821. URL http://doi.acm.org/10.1145/2331812.2331821.

Radia Perlman. Tortis: Toddler's own recursive turgle interpreter system. 10 1974.

Orit Shaer and Eva Hornecker. Tangible user interfaces: Past, present, and future directions. *Foundations and Trends® in Human–Computer Interaction*, 3(1–2):4–137, 2010. ISSN 1551-3955. doi: 10.1561/1100000026. URL http://dx.doi.org/10.1561/1100000026.

S. Subramanian, D. Pinelle, J. Korst, and V. Buil. Tabletop collaboration through tangible interactions. In *16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2007)*, pages 412–417, June 2007. doi: 10.1109/WETICE.2007.4407200.

Hideyuki Suzuki and Hiroshi Kato. Interaction-level support for collaborative learning: Algoblock-an open programming language. In *The First International Conference on Computer Support for Collaborative Learning*, CSCL '95, pages 349–355, Hillsdale, NJ, USA, 1995. L. Erlbaum Associates Inc. ISBN 0-8058-2243-7. doi: 10.3115/222020.222828. URL https://doi.org/10.3115/222020.222828.

Brygg Ullmer, Hiroshi Ishii, and Dylan Glas. Mediablocks: Physical containers, transports, and controls for online media. 09 1998. doi: 10.1145/280814.280940.

John Underkoffler and Hiroshi Ishii. Urp: A luminous-tangible workbench for urban planning and design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '99, pages 386–393, New York, NY, USA, 1999. ACM. ISBN 0-201-48559-1. doi: 10.1145/302979.303114. URL http://doi.acm.org/10.1145/302979.303114.

Simon Voelker, Kosuke Nakajima, Christian Thoresen, Yuichi Itoh, Kjell Ivar Øvergård, and Jan Borchers. Pucs: Detecting transparent, passive untouched capacitive widgets on unmodified multi-touch displays. In *Proceedings of the 2013 ACM International Conference on Interactive Tabletops and Surfaces*, ITS '13, pages 101–104, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2271-3. doi: 10.1145/2512349.2512791. URL http://doi.acm.org/10.1145/2512349.2512791.

Simon Voelker, Christian Cherek, Jan Thar, Thorsten Karrer, Christian Thoresen, Kjell Ivar Øvergård, and Jan Borchers. Percs: Persistently trackable tangibles on capacitive multi-touch displays. In *UIST '15: Proceedings of the 28th Annual ACM Symposium on User Interface Software and Technology*, UIST '15, pages 351–356, New York, NY, USA, November 2015. ACM. doi:

10.1145/2807442.2807466. URL http://dx.doi.org/10.1145/2807442.2807466.

K.N. Whitley. Visual programming languages and the empirical evidence for and against. *Journal of Visual Languages & Computing*, 8(1):109 – 142, 1997. ISSN 1045-926X. doi: https://doi.org/10.1006/jvlc.1996.0030. URL http://www.sciencedirect.com/science/article/pii/S1045926X96900300.

Index

Typeset January 27, 2020