RWTH AACHEN UNIVERSITY

Grippo: Using Grip Gestures to Repurpose Everyday Objects as Controllers

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

by
Tatiana Smirnova

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr. Martina Ziefle

Registration date: 02.09.2014
Submission date: 25.02.2015

I hereby declare that I have created this work completely on my own and used no other sources or tools than the ones listed, and that I have marked any citations accordingly.

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt sowie Zitate kenntlich gemacht habe.

Aachen, February 2015
Tatiana Smirnova

Contents

Abstract **xix**

Acknowledgements **xxi**

Conventions **xxiii**

1 Introduction **1**

1.1 Programming Everyday Objects as Controllers Using Grip Gestures 2

1.2 Research Questions 4

1.3 Outline 5

2 Related Work **7**

2.1 Everyday Objects as Controllers 7

2.1.1 WorldKit: Xiao et al., 2013 8

2.1.2 OnObject: Chung et al., 2010 10

2.1.3 Jabberstamp: Raffle et al., 2007 11

2.1.4 I/O Brush: Ryokai et al., 2004 13

2.1.5 Building Upon Everyday Play: Zhang et al., 2007 14

2.1.6 Opportunistic Music: Hachet et al., 2009 16

2.1.7 Smarter Objects: Heun et al., 2013 . . 17

2.1.8 iCon: Cheng et al., 2010 18

2.1.9 Instant User Interfaces: Corsten et al., 2013 20

2.1.10 Summary: Everyday Objects as Controllers 23

2.2 Programming by Demonstration 26

2.2.1 User-Centered Programming by Demonstration - Stylistic Elements of Behavior: Young et al., 2013 26

2.2.2 Exemplar: Hartmann et al., 2007 . . . 28

2.2.3 Touch and Activate: Ono et al., 2013 . 31

2.2.4 Summary: Programming by Demonstration 32

2.3 Grasping Gestures 34

2.3.1 The Prehensile Movements of Human Hand: Napier, 1956 34

2.3.2 Grasp Sensing for Human-Computer Interaction: Wimmer, 2011 36

2.3.3 Summary: Grasping Gestures 38

3 Preparation Phase **39**

3.1 Brainstorming 39

3.2 Prototyping 43

4 Preliminary Study: a Survey on Grasping Objects of a Cylinder Shape 47

4.1 Description 47

4.2 Procedure 48

4.3 Results 48

4.4 Findings 52

5 Software Prototypes 55

5.1 Tracking Technology 55

5.2 Implemented Grip Gestures 56

5.3 User Action Sequences 57

5.4 System Architecture 60

5.5 Data Communication 62

6 Interactive User Study: Programming Everyday Objects as Controllers using Grip Gestures 67

6.1 Study Design 67

6.2 Setup 68

6.3 Procedure 68

6.3.1 Scenarios 69

6.3.2 Tasks and instructions 70

6.4 Evaluation Methods 72

6.4.1 Questionnaires 72

6.5 Participants 74
6.6 Evaluation 74
6.6.1 Observation Results 74
6.6.2 Questionnaire Results 76
Questionnaire A 76
Questionnaire B 81
6.6.3 Interview results 88
6.7 Findings 90
6.8 Limitations 92

7 Summary and Future Work 93
7.1 Summary and Contributions 93
7.2 Recommendations for Designing End-User Strategies for Programming Everyday Objects as Controllers 95
7.3 Future Work 98

A Appendix for the Preliminary Study: a Survey on Grasping Objects of a Cylinder Shape 101

B Appendix for the Interactive User Study 109

Bibliography 121

Index 123

List of Figures

1.1 Applying different grip gestures to a bottle could allow reusing one object for multiple controllers. For example, grasping a bottle from the bottom and rotating it as a knob could be used to control TV volume and the top grasp of the same bottle could be used for adjusting screen brightness. 3

2.1 WorldKit system: a user performs a swipe gesture on a table or couch surface and instantiates *interactors* for controlling devices in the living room. 9

2.2 Mapping voice and gestures with the PBD interface of OnObject system: a user takes an object with an attached to it RFID tag, performs the gesture, then presses a button on the device and records her voice. 10

2.3 Mapping sounds and gestures with the GUI interface of OnObject system. 11

2.4 Jabberstamp: setup consisted of recording tool (Jabberstamp), playback tool (trumpet), microphone and Java application for processing and mapping the input. 12

2.5 I/O Brush allows users to ”pick up” different properties of an object, such as color, texture or movement and apply it on the canvas. 13

2.6 Building upon everyday play: users can turn an everyday object into a game controller by attaching a special wireless clamp to it. 15

2.7 Opportunistic music: everyday objects such as office lamp, sticky notes, staples box and books can be used to create instant widgets for music composition. 16

2.8 Smarter Objects: modifying the behavior of a physical object with an AR application. A user holds an iPad with the AR application in front of the object to identify it, then a respective interface appears on the screen and the user can choose the desired functionality. 18

2.9 iCon GUI: when a user is mapping a controller to an object, a colored bubble representing the detected position of the object is shown on the screen. Then the user could move the object (by this moving the bubble in the GUI) to the desired operation, wait for a "Bind?" sign to appear and then use the mouse click to confirm the mapping. 19

2.10 Instant User Interfaces: in order to repurpose a ball pen as a controller with the GUI interface a user pointed with the pen at the left side of the screen where physical components are presented, he hovered over a GUI element for two seconds to select it, then moved the object in the mid air therefore moving the cursor until he reached the desired control - advancing to the next slide. . . 21

2.11 Instant User Interfaces: results of the evaluation have shown that the GUI approach was preferred by users in the presentation scenario, followed by programming by demonstration and speech control. In the light scenario, speech control had the preference followed by programming by demonstration. The GUI was rated worse due to the inconvenience of pointing at the screen with an object as heavy as a mug. 23

2.12 Summary of the related work 25

2.13 Programming a robot's behavior by demonstration 27

2.14 Exemplar system recognizes the input from the sensor and visualizes it, a user then can set the threshold or filter for recognizing this particular interaction and map it to a continuous or discrete event. 29

2.15 Exemplar: evaluation results show that according to users' feedback Exemplar decreases the time needed for creating prototypes with sensor-based input and that the system motivated and let users experiment more 30

2.16 Mapping an area on the LEGO block to the *Play* controller: training the system by demonstration is done by touching the area on the object and pressing the key on the keyboard associated with a controller. 32

2.17 a) A person uses power grip to unscrew a tightly screwed-up lid of a jar, b) a person changes the grip to a precision grip as the lid loosens. 35

2.18 The GRASP Model includes five factors influencing a grasp: goal, relationship, anatomy, setting and properties of an object. 36

3.1 If a user simply points at a device, without invoking an additional interface, it is not clear which functionality he wants to map to an object – in case of a TV it could be power, volume, navigation between channels, etc. . 41

3.2 Mapping the positions of the mug to the minimum and maximum volume values: a) with a GUI-based approach by manipulating a virtual object on the screen, b) with the PBD approach by setting the real mug in a desired state in front of the smartphone camera. . . . 44

3.3 Using an object without visible state indicators, such as a glass: a) a user, first, selects an object, and then, b) demonstrates the grasp we would like to map to the selected controller without indicating object positions for the minimum and maximum values. 45

4.1 Users' choices of a grip gesture for a single controller (the number on the glass is the number of fingers in a grip). Most common ways to grasp an object of a cylinder shape appear to be with the whole hand from the side or with a four finger grasp from the top of the object. 49

4.2 Users' choices of grip gestures for two controllers show that it is preferable to grip an object on different areas compared to using different number of fingers in a grip. 50

4.3 Users prefer to distinguish between different areas of an object compared to using a different number of fingers in a grip. Moreover, if an object is to small for defining three clear areas for grips, different actions can be performed with an object for representing different controllers. 50

4.4 Users' choices of grip gestures for controlling two degrees of one function did not show clear preference and need to be investigated further. 51

4.5 Participants did not show a preference as a group for using an everyday object with or without a push affordance for a binary controller. 52

4.6 Ten participants preferred to reuse an object for multiple controllers and three participants chose to take a new object with a push affordance. 52

5.1 Two objects selected for the user study were augmented with reflective markers. When a marker is seen by at least two cameras, it can be tracked by the Vicon Nexus software. . . . 56

5.2 Three types of grip gestures were defined on the bottle object: A – top grip, B – middle grip, C – bottom grip, and a tap on top of the bottle (D) was also available for mappings. . 57

5.3 Three touch-sensitive areas were defined on the chocolate bar object and were available for mappings. 57

5.4 User action sequences: a) GUI-based approach, b) PBD approach. Both prototypes included the same sequence of actions for creating a mapping. The only difference was the way a grip gesture is selected or demonstrated. 58

5.5 Interface for initiating the interaction, selecting the device and the controller was the same for the PBD and the GUI-based applications. 58

5.6 PBD application: once a user has identified an object to the system by scanning a QR code attached to it, the system presents which controllers are currently assigned to the object and to which areas (in this example, a TV volume controller is assigned to the bottom area or bottom grasp of the object). Then a user can demonstrate the grip gesture that he would like to map to the controller selected in the previous step (in this Figure middle grasp is being assigned to the TV Screen brightness control). The area corresponding to the performed grip gesture is highlighted on the screen (picture b). The interaction ends with a confirmation screen (picture c) after a user selected the grip gesture and taped on the screen. 59

5.7 GUI-based application: once a user has identified the object to the system, the application presents the mappings that are currently assigned to the object (similarly to the PBD approach), then a user can tap on one of the icons to select a grip gesture that he would like to map to the selected controller (picture b). The interaction ends with a confirmation screen as in PBD application (picture c). . . . 60

5.8 Overview of the system architecture: the Server processes user input from the mobile application and it receives and interprets data stream from the Vicon tracking system. When the mappings are created, the Server application sends translated values to the TV and Lights applications. 61

5.9 Data communication in PBD and GUI-based applications: after the user selects the object, the Server checks if any mappings are currently assigned to it. If none of the controllers are currently assigned to the object, the user is provided with the interface of selecting or demonstrating a grip gesture, and if existing mappings are found, they are presented to the user in the mobile application and then the user proceeds to assigning a grip gesture. 63

5.10 Data communication in the *demonstration mode*: the Server is listening to the data stream from the Vicon system, interprets it and later maps it with the selected controller. 64

5.11 Data communication in the *use mode*: the Server listens to the data from the Vicon system. If a match for the performed grip gesture is found, Server translates the coordinates data into the values for the controllers and sends it to the TV or Lights application respectively. 65

6.1 The setup included eight Vicon Bonita cameras, objects with reflective markers, iPhone applications for the PBD and GUI approaches, TV application and Phillips Hue smart lights. User's hand was also augmented with three reflective markers. 69

6.2 Users found the features of the GUI-based and PBD applications comprehensive. 77

6.3 Users rated the GUI-based and PBD applications as easy to use. 78

6.4 Both GUI-based and PBD applications allowed users to express their intentions. . . . 79

6.5 Visualization of previously assigned to an object grasps and controllers was clear and helpful to users in both GUI-based and PBD approaches. 80

6.6 Results show that users found it easier to identify available grasps with the GUI-based prototype. 82

6.7 Nine users agreed that applying the grasp after programming the object with the PBD felt more natural compared to the GUI-based application. 83

6.8 When users were asked if they found PBD harder compared to the GUI-based approach, only six users agreed with the statement. The remaining eight users either disagreed that PBD interaction was harder (four participants), or did not agree or disagree with the statement. 84

6.9 Eight users found the PBD approach more time-consuming compared to the GUI-based approach. One of the reasons was the need to switch the attention between the smartphone and demonstration on the object. 85

6.10 Switching attention between the smartphone and demonstration on the real object was found interruptive by eight users. 86

6.11 Users found it feasible to map grasps or motions with more degrees of freedom with both the PBD approach (12 users) and the GUI-based application (10 users). 87

6.12 Results of users' preference between the GUI-based and PBD approaches for assigning gestures with more degrees of freedom. . 87

List of Tables

6.1 Tasks and instructions for the user - TV control 71

6.2 Tasks and instructions for the user - Lights control 71

Abstract

In cases when a dedicated object is missing, a similar object could be appropriated to temporary substitute it. For instance, a knife could be used to open an envelope in the absence of a special letter opener. We applied this principle of repurposing an object to the domain of digital interfaces and saw that everyday objects could be used to temporary substitute a dedicated controlling device, such as a remote control or a light switch. Since everyday objects have similar physical properties as dedicated controllers (e.g., buttons, knobs), they provide haptic feedback, therefore supporting eyes-free interaction. Everyday objects are already integrated in the user's environment and are easily accessible. Typical controlling devices provide multiple functions, for instance, a TV remote control allows to adjust volume, switch between channels, program favorite channels, etc., therefore in this thesis we present a way of assigning multiple controllers to one object by using different grip gestures in analogy to traditional multi-functional controlling devices. So far, conducted research has not evaluated end-user strategies of communicating the desired system's behavior and programming objects as controllers using grip gestures. Thus, the aim of this work is to conduct user studies and investigate these strategies. In particular, the thesis focuses on Programming by Demonstration (PBD). PBD is inspired by research in robotics, where a user can teach a system certain behavior by demonstrating it. Moreover, PBD can help novice users and users without programming knowledge to communicate their intentions. In order to collect the data, prototype systems representing PBD and GUI-based end-user programming strategies will be implemented; afterwards qualitative studies evaluating user's perception and feedback will be conducted. This work ought to provide clearer understanding of user preferences in the domain, and the results may serve as recommendations for designing interactive systems using PBD for repurposing everyday objects as controllers.

Acknowledgements

I would like to thank all the users who participated in my studies for their time. Your valuable feedback made a big part of this work, thank you for that.

I want to say thank you to my supervisor – Christian Corsten, M.Sc., who always provided help when I needed it. Christian, I enjoyed working with you and I have learned a lot from you, thank you very much!

I would like to thank Prof. Dr. Jan Borchers who created such a great working environment at the chair, it was a pleasure to work at i10 every day, thank you!

I would also like to thank Prof. Dr. Martina Ziefle for her support and being responsive despite having so much research work.

And last, but not least, I would like to thank my friends and family for their patience and support when I needed it. It would not be possible without you!

Thank you very much,
Tatiana

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

The whole thesis is written in American English.

Chapter 1

Introduction

In everyday life we often appropriate objects similar to the dedicated ones to temporary substitute them. For instance, we use a cutlery knife to open an envelope if we do not have a dedicated tool for this goal – a letter opener. Both objects have similar physical properties and allow us to open a letter. Appropriating objects is convenient, since we can use everyday objects in our vicinity to substitute a dedicated tool. Appropriation or repurposing an object lessens the dependency on the dedicated tools and devices.

Objects with similar to the dedicated tools physical properties can temporary substitute them.

We applied the idea of repurposing objects to the domain of digital interfaces. In situations where dedicated controlling devices, such as a TV remote control, are unavailable or out of reach, everyday objects can be exploited to temporary substitute them. Everyday objects are located in user's vicinity and they are easily accessible. Spontaneous access to objects provides a quick replacement for the original controller. Moreover, physical affordances of objects could provide haptic feedback and support eyes-free interaction due to tactile clues. Everyday objects can be repurposed as controllers for reasons of convenience to substitute an unavailable or broken controller, for example, if a TV remote controller is out of service since it has low batteries, and a user wants to adjust the volume – a water bottle could be repurposed for controlling volume by rotating it as a knob. Another reason for repurposing an everyday object is to create a duplicate of the original controller if it is located

Repurposing everyday objects as controllers could make interaction with digital devices more convenient.

at a certain distance from the user, e.g., a user could push a button on a ball pen to turn on and off the lights and temporary substitute a light switch located on another side of the room. Everyday objects can also serve as short cuts for a sequence of actions, such as opening a specific webpage in browser. Repurposing everyday objects as controllers provides quick access to controlling digital devices.

1.1 Programming Everyday Objects as Controllers Using Grip Gestures

We investigate programming everyday objects as controllers using grip gestures.

Typical dedicated controllers provide multiple functions in one device. For example, with a TV remote control users can navigate to a next and previous channel, program favorite channels, adjust volume and screen brightness, etc.. Motivated by this fact, in this thesis we extend the idea of repurposing an everyday object as a controller by adding the possibility of using different grip gestures for mapping controllers to an object. By applying different grip gestures users could assign multiple controllers to one object, which saves space as it minimizes a necessary number of objects if multiple controllers are to be mapped. For example, grasping a bottle from the bottom and rotating it as a knob could be used to control TV volume and the top grasp of the same bottle could be used for adjusting screen brightness (Figure 1.1). A tap on top of the bottle could be used, for instance, to switch a TV on and off or to mute and unmute the volume.

Programming an object as a controller requires two components: technology and interaction model.

In order to make an everyday object interactive and repurpose it as a controller two components are required: the *technology* and the *interaction model*. *Technology* needs to provide stable and robust tracking of an object and grip gestures, it could be vision or sensor-based, several alternatives have been investigated in the existing research, for example research by Penelle and Debeir [2014]. *Interaction model* implies how users convey their intentions of repurposing an object as a controller. This thesis focuses on the interaction component and how end users can program or "teach" objects to act as controllers by using grip gestures.

Figure 1.1: Applying different grip gestures to a bottle could allow reusing one object for multiple controllers. For example, grasping a bottle from the bottom and rotating it as a knob could be used to control TV volume and the top grasp of the same bottle could be used for adjusting screen brightness.

GUI and PBD are the most common strategies for mapping controllers to everyday objects.

End-user programming makes a system customizable according to user's needs and desires. Currently, the most common strategies for assigning the functionality are: interaction via Graphical User Interface (GUI), such as *iCon* system by Cheng et al. [2010] and demonstration of the desired behavior – Programming by Demonstration (PBD), for example, *OnObject* system by Chung et al. [2010]. Although interaction via GUI could be more familiar to the user, PBD could be more intuitive as it allows direct interaction with the object, which could be an advantage for demonstration of grip gestures.

We compare GUI and PBD approaches regarding qualitative characteristics.

Existing research has not yet covered comparison of PBD with other end-user programming strategies, such as a GUI. Thus, the contribution of the thesis is to conduct this comparison and investigate a PBD approach in more details regarding various qualitative characteristics. PBD is inspired by research in robotics, where a user can teach a system certain behavior by demonstrating it, and we would like to translate this paradigm to the domain of programming everyday objects. A PBD approach could be appro-

Our PBD approach is inspired by a demonstrational programming in robotics.

priate for novice users and users without programming knowledge, as it allows to teach a system, or, in our case, an object, to act in a certain way by demonstration. Demonstration could be a more convenient way to assign grip gestures to an object, as it allows users to interact with an object directly, compared to interaction via a virtual GUI.

Findings of the conducted study are summarized in recommendations for designers.

In this thesis we conduct a qualitative user study with implemented software prototypes representing GUI and PBD strategies. We analyze users' preferences for programming everyday objects using grip gestures and summarize our findings in a form of recommendations for designing systems that implement end-user strategies for programming everyday objects.

1.2 Research Questions

The main goal of this work was to investigate a *Programming by Demonstration (PBD)* approach as an end-user strategy for programming everyday objects as controllers using grip gestures. We will compare this approach to another common way of mapping everyday objects and controllers – programming via a *Graphical User Interface (GUI)*. The main research questions of this thesis are:

- Is *Programming by Demonstration* preferred over the *GUI-based approach* for programming everyday objects using grip gestures?
- What are the benefits and limitations of programming everyday objects by demonstration?
- In which scenarios can a GUI-based approach be more appropriate than PBD?

1.3 Outline

The thesis is structured in the following way:

- **Chapter 2** presents the review of the related work in the field of using and programming everyday objects as controllers. We also looked into existing research regarding programming by demonstration paradigm and the research related to grasping gestures and their use in HCI.
- **Chapter 3** describes the preparation process of designing and prototyping software applications representing the PBD and GUI-based approaches for the interactive user study.
- **Chapter 4** is dedicated to the preliminary survey study on grasping objects of a cylinder shape.
- **Chapter 5** describes the technology and implementation of software prototypes.
- **Chapter 6** presents the interactive user study and conducted evaluation of presented approaches for programming everyday objects using grip gestures.
- **Chapter 7** concludes the thesis with a summary, contribution and description of future work. In this chapter, we also summarize our findings in a form of design recommendations for systems that implement end-user programming of everyday objects.

Chapter 2

Related Work

During the literature review we examined the projects that use *everyday objects as controllers* to see which end-user programming strategies have been implemented and evaluated.

Then we investigated the paradigm of *Programming by Demonstration* in more details to identify in which application areas it has been used and what are its known benefits.

Since Grippo is based on using grip gestures for programming everyday object, at the end of the chapter we examine the literature regarding *grasping gestures* and its classifications.

We examined the domain of using everyday objects as controllers, the paradigm of *Programming by Demonstration* and we conducted literature research regarding applying knowledge of grasping gestures in HCI.

2.1 Everyday Objects as Controllers

Interaction with everyday objects as controllers has been investigated in the existing research. Further in this chapter we will examine implementations of systems that enable using everyday objects as controllers based on the following criteria:

- Application area

  We looked at the application domains where everyday objects have been used as controllers and observed in which applications there was a need for an end-user programming interface.

- Presence of an end-user programming interface

  We investigated what kind of end-user programming strategies have been implemented in the existing research. We paid special attention to the systems that used *Programming by Demonstration* to see the benefits and possible limitations of the approach.

- Technology

  We observed the technical solutions implemented in the described systems.

- Use of affordances

  We noted those applications, where the affordances of the physical objects were exploited.

- Performed evaluation

  We examined what kind of evaluation was performed for the systems in this domain, most importantly for us was whether an evaluation of the end-user programming approach has been performed.

2.1.1 WorldKit: Xiao et al., 2013

WorldKit allows to make everyday surfaces interactive.

WorldKit by Xiao et al. [2013] is a system that turns everyday surfaces into interactive interfaces. Users can invoke an *interactor* - a projected widget - by a hand gesture (e.g. swipe on the surface), by voice control or using a smartphone application. For instance, a user can invoke an *interactor* on a coffee table or a couch by a swipe gesture, when the gesture is recognized by the system the instantiated controllers for operating a home entertainment system will be projected on the respective surfaces (Figure 2.1).

Figure 2.1: WorldKit system: a user performs a swipe gesture on a table or couch surface and instantiates *interactors* for controlling devices in the living room.

However, users do not have control over the choice of *interactors* that can be triggered by their actions. The content of the application and its instantiated controllers have to be predefined by the designers. This is a limitation of the system, as there is no end-user programming interface, system can only be modified by programmers. Xiao et al. [2013] created a framework for programming applications based on WorldKit, that includes a library of interactors. Based on the provided abstractions developers can implement e.g., a *binary contact interactor*, that detects touch, or a *counting interactor*, that counts the number of items (contact blobs) in the area. Examples of the WorldKit applications include interactions in the living room for substituting dedicated devices (e.g., a remote control) or an office scenario where a user can set an automatic notification about being in a meeting when the office door is closed. WorldKit can also provide an assistance during cooking for measuring the ingredients and projecting the progress of following a recipe.

WorldKit does not provide an end-user programming interface, the mappings are predefined by the designers of an application.

- WorldKit allows users to invoke instant user interfaces projected on everyday surfaces and demonstrates various application scenarios that can be implemented
- WorldKit enables *developers* to implement interactive applications on everyday surfaces, however, end-user programming, that was not presented by authors of the system, could enhance the system and give users power in creating mappings, thus making the application more flexible

2.1.2 OnObject: Chung et al., 2010

OnObject is an application that provides interactive play and storytelling for children.

Chung et al. [2010] presented an OnObject system, that can be used for interactive storytelling or game-based learning applications for children through linking gestures and sound to objects. The system consists of portable RFID reader with integrated microphone and speaker; the gadget can capture and recognize gestures based on acceleration, such as shaking, swinging, thrusting, tilting and applying circular motion.

OnObject system has two end-user programming approaches: PBD and desktop GUI.

OnObject has two end-user programming interfaces: Programming by demonstration and the desktop GUI application. With a simple PBD approach users, even small children, can map voice sounds to a gesture performed with an object in a following way: the user wears a small RFID reader device on his hand and takes the object with an attached to it RFID tag, he performs the gesture, then presses a button on the device and records his voice (Figure 2.2).

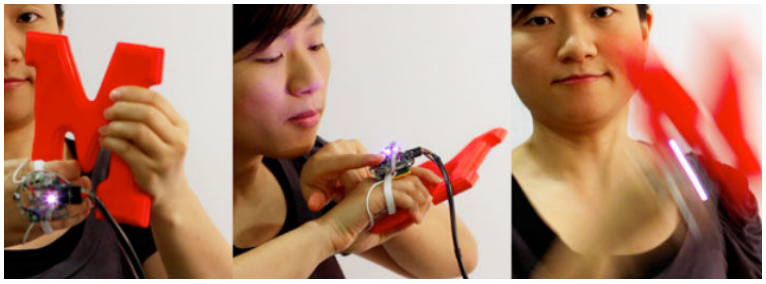


Figure 2.2: Mapping voice and gestures with the PBD interface of OnObject system: a user takes an object with an attached to it RFID tag, performs the gesture, then presses a button on the device and records her voice.

Desktop GUI allows mapping sounds that are hard to be reproduced by voice, such as sword thrust.

GUI-based desktop application is more complex, but also more flexible: it is not suitable for children, however, parents or teachers can map special sounds, that are hard to be produced by voice and create interactive stories for children (Figure 2.3).

- OnObject allows the user to assign mappings to *motions* performed with an object, whereas Grippo lets

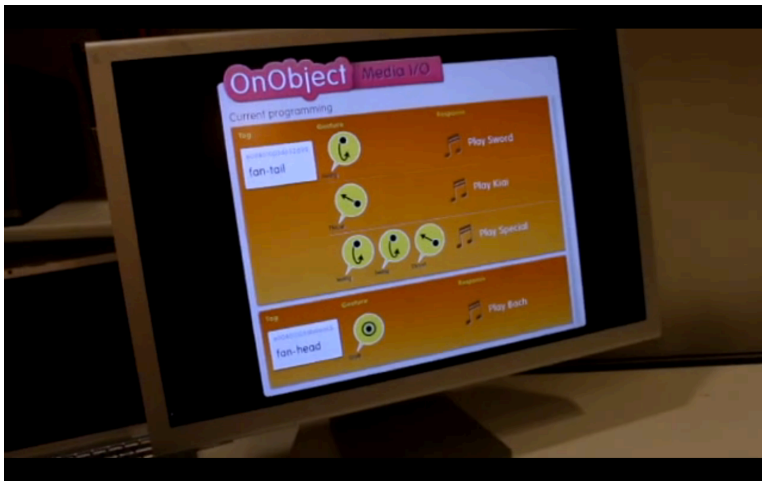


Figure 2.3: Mapping sounds and gestures with the GUI interface of OnObject system.

the user apply different *grip gestures* and respectively map several controllers to an object.

- OnObject and Grippo have different application areas: OnObject is designed for interactions with children, such as storytelling and interactive play, in Grippo we implemented controllers for the domestic scenarios, such as controlling devices in the living room.
- An important characteristic of PBD implemented in the OnObject system is that it is suitable for small children unlike the GUI approach: it is more natural and straightforward for kids to perform a gesture and record a voice, than to select an icon of the gesture and link it with the sound in the computer interface.

2.1.3 Jabberstamp: Raffle et al., 2007

Jabberstamp is an interactive application for children that allows mapping sound to children's drawings.

Raffle et al. [2007] presented Jabberstamp: a system that allows children to synthesize their drawings with voices. The system is similar to the OnObject by Chung et al. [2010] in a way that it enables easy to use interaction for children. If with OnObject users were able to map their voices with gestures performed with objects, with Jabberstamp children

could map their voices to their drawings. First, a child created a drawing on a piece of paper, then he put the paper on a special tablet and pressed a button on the tablet to identify the drawing. In order to record a sound children took a *Jabberstamp* - special ink-based rubber stamp integrated with a microphone, pressed the stamp on the page and recorded the sound. After finishing the sound recording they released the stamp, which left a red star on the paper. Next, children could listen to their recordings by touching the red star with a special trumpet (the tools are shown on the Figure 2.4).

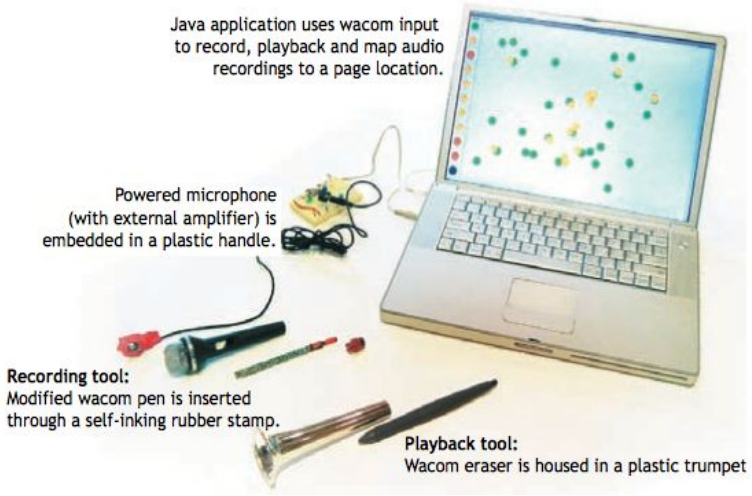


Figure 2.4: Jabberstamp: setup consisted of recording tool (Jabberstamp), playback tool (trumpet), microphone and Java application for processing and mapping the input.

PBD is used for mapping voice to a specific part of a drawing.

Jabberstamp system uses programming by demonstration for mapping a voice with a picture: a user demonstrated the system *what to map* (a part of a drawing) to which sound. Raffle et al. [2007] conducted a study with children, that has shown that the application was rather quickly understood by a 4-8 years old children after an explanation of what the system does. Jabberstamp allows children to develop creative skills as well as literacy skills in a form of a game. Moreover, authors emphasize that children's use of *paper and tangible tools* (rather than a purely digital approach) allowed the system to become *"an intuitive extension to children's knowledge of the environment"*.

- Jabberstamp has shown that programming by demonstration can be effectively used for interactive applications for children: PBD enables *direct interaction* with an object (a drawing) and *tangible tools*, that made the application comprehensive and easy to use by children

2.1.4 I/O Brush: Ryokai et al., 2004

With I/O Brush users can copy or "demonstrate" object's properties and apply it on a canvas during drawing.

I/O Brush by Ryokai et al. [2004] is a drawing tool for young children that uses everyday objects as ink. The system consists of a canvas (Wacom tablet) and electronic brush. The brush has an embedded camera, two spring-based touch sensors, supplemental lights for the camera and optical fibers for indicating when the brush has "picked up" an object's attribute. During interaction the child first uses the brush to collect color, texture or even movement of an object, then as he moves the brush across the canvas the respective object's property is applied (Figure 2.5).

Figure 2.5: I/O Brush allows users to "pick up" different properties of an object, such as color, texture or movement and apply it on the canvas.

Using PBD allowed children to explore objects and the environment around them.

The evaluation of I/O Brush has shown that children learned how to use the system quickly. I/O Brush motivated to explore the environment and objects' properties, the demonstrational approach here served a learning purpose.

- I/O Brush presents a new application of programming by demonstration, i.e. not only gestures with objects or object's areas can be used for mappings (as in the systems described above), but also properties of objects, such as color or texture. This can be used in domestic interactions as well, for instance, by scanning an object users can "copy" its color and apply it to the color of the lights in the room. In the current implementation of Grippo this interaction is not covered due to a different scope, but the system could be extended in future work.
- I/O Brush concept supports the idea of using PBD for interactive application for children. One of the reasons is that PBD allows to explore and use the affordances of physical objects.

2.1.5 Building Upon Everyday Play: Zhang et al., 2007

PBD can be used for programming everyday objects as game controllers.

"Building upon everyday play" by Zhang and Hartmann [2007] allows to turn everyday objects into game controllers for a pervasive game experience. The system consists of a motion sensing clamp, that can be attached to an everyday object (Figure 2.6), and an application based on the Exemplar system (described in section 2.2.2 "Exemplar: Hartmann et al., 2007"), that translates sensor data to the game motions. User can create his own game controllers by demonstration: first he attaches the clamp to a desired object (e.g. an office chair), then according to system's instructions user can show the system which motion he would like to use for a certain game action.

- "Building Upon Everyday Play" shows the potential of using PBD in gaming application domain, in

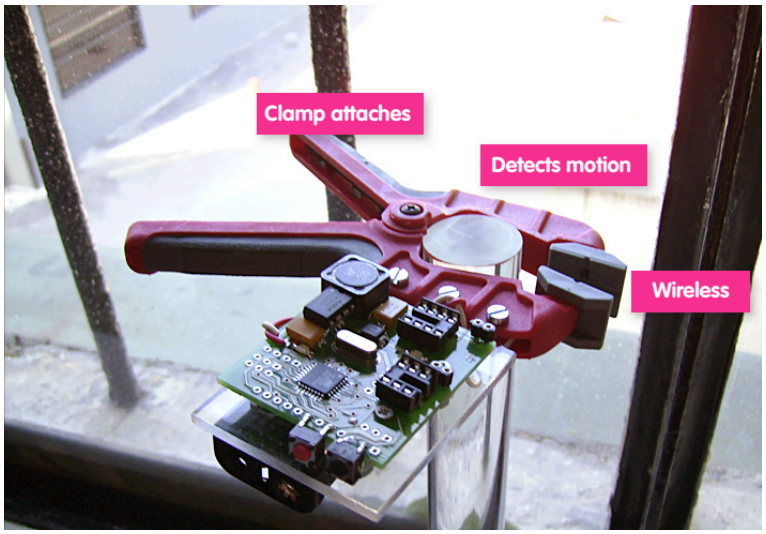


Figure 2.6: Building upon everyday play: users can turn an everyday object into a game controller by attaching a special wireless clamp to it.

Grippo we have implemented PBD for tasks in the domestic environment and interactions with the devices in the living room.

- The system described in "Building Upon Everyday Play" lets users map free motions as game controllers, Grippo has a defined set of interactions with the object and it recognizes grip gestures applied to an object.

- The interaction metaphor in the "Building Upon Everyday Play" is different from a typical way how everyday objects are mapped to the controllers: when users mapped their motions to the game actions they were mimicking the actual movement (e.g. slide the chair forward to move forward in the game) and not trying to repurpose an object as a dedicated game controller, such as a joystick. This shows how everyday objects can make game experience more exhaustive and that PBD approach is a suitable solution for this paradigm.

Mimicking actual game actions versus simulating a traditional game controller.

2.1.6 Opportunistic Music: Hachet et al., 2009

Everyday objects can be repurposed to become widgets for music composition.

Opportunistic music by Hachet et al. [2009] allows to recreate musical controllers from user's immediate environment. Authors show how office supplies can be turned into widgets for music control: an edge of the office lamp can be used as a frequency oscillator, trigger of the drum loop can be mapped to a tap on top of the sticky notes and a volume fader can be created by sliding a staples box along the book edge (Figure 2.7).

Figure 2.7: Opportunistic music: everyday objects such as office lamp, sticky notes, staples box and books can be used to create instant widgets for music composition.

PBD is used for mapping objects with music controllers.

Opportunistic music uses two cameras for stereovision-based 3D positioning an a FSR sensor on user's finger. Users can assign mappings by demonstration: first, a user selects a sound parameter, then he chooses the object he would like to turn into a widget, then he applies some pressure on the desired surface and cameras track the starting position of the widget, then the user continues sliding along the surface until he reaches the desired size of the controller and releases his finger.

- Opportunistic music allows to use areas on the objects for mapping the controllers, moreover with the help of a FSR sensor users can set granularity to the created

widgets. With Grippo users could not define the start and the end positions of the created controller, for simplicity users were asked only to demonstrate the grip gesture they wanted to map and values were predefined and interpolated along the 360 degrees area of the object rotation around itself.

- Opportunistic music uses PBD to define widgets, however, no evaluation has been performed; Grippo prototypes were implemented to evaluate PBD and GUI-based end-user programming strategies.

2.1.7 Smarter Objects: Heun et al., 2013

Smarter Objects present an *Augmented Reality* approach for programming physical objects.

Heun et al. [2013] presented Smarter Objects – a system that associates a physical object with it's virtual representation and allows to modify the interface and behavior of that physical object and its interaction with other "smarter objects". Augmented Reality (AR) approach was used in the system to provide an interface for adding new functionality to an object. For instance, if the user wants to program a physical knob for switching between sound tracks, he holds the iPad with the AR application in front of the object to identify it, then a respective interface appears on the screen and the user can choose the desired functionality (Figure 2.8). Moreover, the user can connect "smarter objects" to each other, for example, a radio object can be connected to a speaker object by drawing a line between them in the AR GUI application.

AR approach allows to combine the flexibility of a virtual GUI experience with a haptic feedback of physical objects.

- Smarter Objects presented a paradigm of adding functionality to a physical object with an AR GUI application, thus combining interaction with haptic feedback with the flexibility of a virtual GUI experience. However, the process of programming objects has not been evaluated. Grippo covers the evaluation and comparison of two proposed programming strategies - PBD and the GUI-based approach.
- Smarter objects provided users with the possibility of connecting objects between each other, in Grippo this interaction was not investigated.

Figure 2.8: Smarter Objects: modifying the behavior of a physical object with an AR application. A user holds an iPad with the AR application in front of the object to identify it, then a respective interface appears on the screen and the user can choose the desired functionality.

- Smarter Objects system allows to assign values to different states of a rotary knob or push buttons on the objects that were *designed for the purpose of the study*; in Grippo we focused on repurposing *everyday objects* such as a water bottle or a chocolate box with minimal augmentation. We used the affordances provided in the user's environment, for instance, users can rotate a water bottle as a knob, but there are no clearly defined states or physical constraints, such as minimum and maximum.

2.1.8 iCon: Cheng et al., 2010

iCon system represents a common GUI approach for programming everyday objects

Cheng et al. [2010] present iCon system, that implements an approach for using everyday objects as additional instant desktop controllers. iCon allows users to map to everyday objects binary and consecutive controllers. Implemented binary controllers include bookmarking a web page, saving a current document, copy and paste functionality, toggles between play and pause, mute and unmute function, etc. The list of consecutive controls in iCon consists of such functions as switching to a next and previous song, increas-

ing and decreasing the volume and zoom in and out. As an object for mappings authors of iCon chose a bottle, that has affordances to be grasped by the hand and easily moved.

iCon system focused on using everyday objects as desktop controllers, and end-user programming was done in the setting of user's work desk.

iCon setup consisted of a web camera installed on top of the desk (an eagle-eye view) or under the desk. Objects were augmented with a special sticker (fiducial) to enable the tracking. Users could program an object with a desktop GUI application. First, user attached a sticker to the object, placed it in the detecting area, then in the application user could see a colored bubble representing the detected position of the object. Then the user could move the object (by this moving the bubble in the GUI) to the desired operation, wait for a "Bind?" sign to appear and then use the mouse click to confirm the mapping (Figure 2.9).

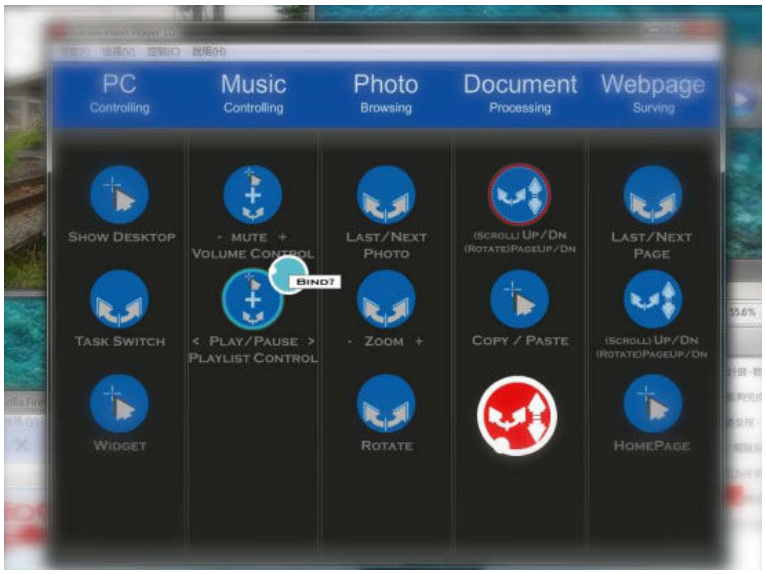


Figure 2.9: iCon GUI: when a user is mapping a controller to an object, a colored bubble representing the detected position of the object is shown on the screen. Then the user could move the object (by this moving the bubble in the GUI) to the desired operation, wait for a "Bind?" sign to appear and then use the mouse click to confirm the mapping.

- Both iCon and Grippo allow to map multiple controllers to an object. With iCon users had three *gestures* that they could apply to an object: click, rotate

and drag; Grippo lets users apply top, middle and bottom *grip gestures* for object rotation and a tap on top of the object, this way up to four controllers can be assigned to an object.

- iCon used a Time-To-Live mechanism to distinguish between the normal and repurposed uses of the object, Grippo allows users to program an area of the object for repurposed use (e.g. grasp the bottle from the top for volume control).

- End-user programming in iCon is presented only with the desktop GUI application; for the Grippo we developed two prototypes for programming everyday objects – GUI-based and PBD.

- Both iCon and Grippo use intuitive mappings with affordances of objects, that are tangible, graspable and movable.

- Evaluation part in iCon was focused on possible scenarios for using the developed system and efficiency of using everyday objects in a context switching scenario; with Grippo we investigated how users can program everyday objects with the GUI-based and PBD approaches.

2.1.9 Instant User Interfaces: Corsten et al., 2013

A preliminary evaluation of a PBD, GUI-based and Speech control end-user programming strategies was conducted by Corsten et al. [2013]

Corsten et al. [2013] introduced a concept of *Instant User Interfaces* and implemented a system for repurposing everyday objects as controllers. The term *Instant UI* is defined as a *"user interface that lets a user select a set of arbitrary physical objects within reach to instantiate it as controller for a technical system"*. Authors conducted a preliminary evaluation of the end-user programming strategies, that enable a user to *instantly* establish mappings between objects and e.g. digital devices. The following end-user programming strategies were included in the evaluation: *Graphical User Interface, Programming by Demonstration* and *Speech Control*.

Sample tasks during the study included presentation control with a ball pen and lights control with a mug. An example of programming a ball pen with a GUI approach is shown on Figure 2.10: user points the object at the left part of the screen where physical components are presented, he hovers over a GUI element for two seconds to select it, then he moves the object in the mid air therefore moving the cursor until he reaches the desired control - advancing to the next slide. By pointing *an object* at the screen user identifies it to the system without additional steps of selecting an object.

A GUI-based programming approach included pointing the object at the desktop GUI.

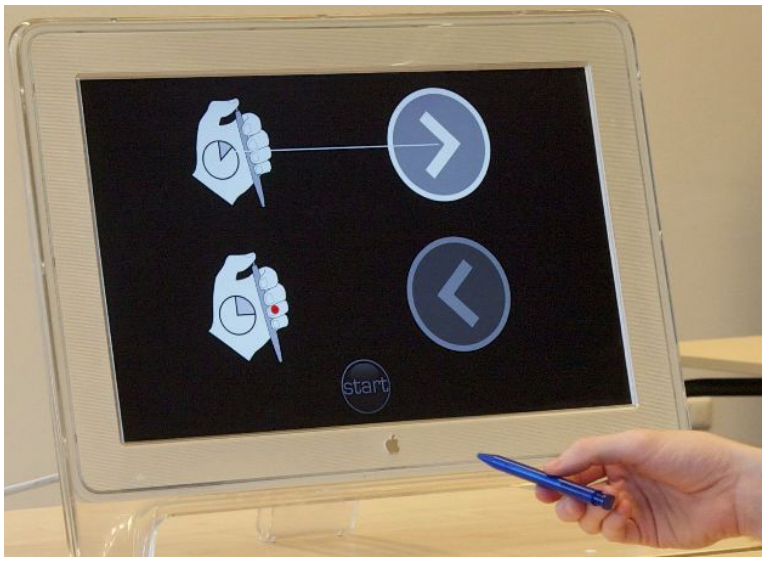


Figure 2.10: Instant User Interfaces: in order to repurpose a ball pen as a controller with the GUI interface a user pointed with the pen at the left side of the screen where physical components are presented, he hovered over a GUI element for two seconds to select it, then moved the object in the mid air therefore moving the cursor until he reached the desired control - advancing to the next slide.

For programming an object by demonstration a user had to trigger both events simultaneously, i.e., in the scenario with a ball pen a user would push the pen button and press right-arrow key on the keyboard (as a shortcut for advancing to the next slide) to establish the mappings.

During PBD users needed to trigger the controller and perform an action with an object simultaneously.

Speech control for programming everyday objects was presented in a form of a Wizard-of-Oz study. Users were not

limited by a vocabulary and could user natural language for creating the mappings.

Users experienced problems with simultaneous interaction with a controller and an object located on different planes during PBD.

Evaluation of the described programming strategies has not shown a clear preference, but provided some insights of user experience. For example, it was found that simultaneous rotation of a knob mounted on the wall (vertical plane) and the mug standing on the table is problematic (horizontal plane), therefore the PBD implementation was not chosen an optimal strategy for assigning continuous controls. However, users did not experience any problems with mapping a push of a pen button (a binary control) for presentation control. In fact programming a pen was considered easy with all presented approaches.

Pointing an object at the GUI was found inconvenient if the object is relatively heavy.

The GUI approach was preferred by users in the presentation scenario (M = 5.25), followed by programming by demonstration (M = 4.92) and speech control (M = 4.83). In the light scenario, speech control had the preference (M = 5.83) followed by programming by demonstration (M = 5.67). The GUI was rated worse (M = 3.50) (Figure 2.11), the possible reason given by authors was the inconvenience of pointing at the screen with an object as heavy as a mug.

- Corsten et al. [2013] have performed a preliminary evaluation of end-user strategies for programming everyday objects. With Grippo we extend this work and evaluate two approaches of assigning controllers to an object using *grip gestures* that was not covered in the described paper.
- Implementation of PBD by Corsten et al. [2013] can be effective for some tasks, however, in Grippo we designed another approach for programming everyday objects by demonstration. One of the reasons was that simultaneous trigger of a controller and an action with an object can be problematic, for instance, if they are at a certain distance from each other. Moreover, this approach would require an additional vocabulary for deleting and correcting the mappings. In order to make PBD approach applicable for a wider range of tasks we included a smartphone application in the interaction.

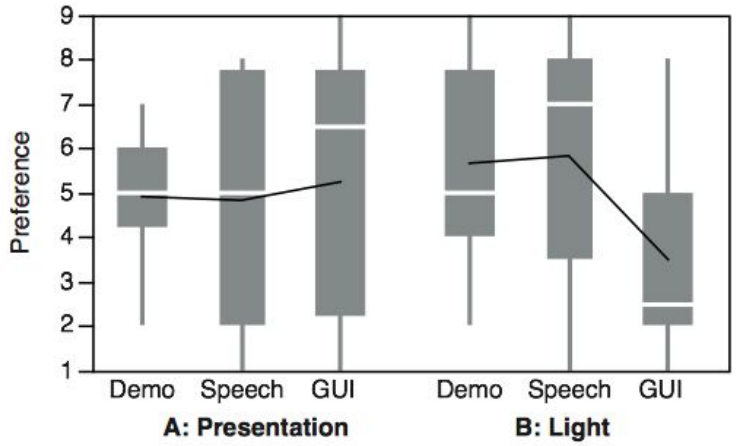


Figure 2.11: Instant User Interfaces: results of the evaluation have shown that the GUI approach was preferred by users in the presentation scenario, followed by programming by demonstration and speech control. In the light scenario, speech control had the preference followed by programming by demonstration. The GUI was rated worse due to the inconvenience of pointing at the screen with an object as heavy as a mug.

2.1.10 Summary: Everyday Objects as Controllers

The paradigm of using everyday objects as controllers has been applied in various domains.

In the section 2.1 "Everyday Objects as Controllers" we presented our review of systems that enable using everyday objects as controllers. The summary of the related work is presented in Figure 2.12. Everyday objects can be repurposed to provide instant control via interactive surfaces (Xiao et al. [2013]) or to create a mobile setup for music composition (Hachet et al. [2009]). It can be used in applications for children, such as interactive storytelling (Chung et al. [2010]) or drawing (Raffle et al. [2007]), or as instant game controllers to make the experience more exhaustive (Zhang and Hartmann [2007]). Another common application area for using objects as instant controllers is the domain of everyday tasks in office and domestic environments (Corsten et al. [2013], Cheng et al. [2010], Heun et al. [2013]).

End-user programming adds flexibility to the system by allowing the user to set the mappings between objects and controllers.

During the literature review we focused on the systems that provide end-user programming capabilities, only one project from the presented literature – WorldKit – did not have end-user programming interface. WorldKit system by Xiao et al. [2013] had a framework for developers, who have the control of application mappings. End-user programming could add flexibility to the system and give end users control over mappings and invoked interfaces.

Evaluation of the described systems focused mostly on the interaction part of using an object as a controller, but not on the end-user programming part.

We identified the following end-user strategies for programming everyday objects: *Graphical User Interface (GUI), Programming by Demonstration, Augmented Reality (AR)* and *Speech control*. We also saw that, although most of the systems have been evaluated through user studies, those evaluations focused on the interaction of *using* an object as a controller, but the evaluation of the *programming* an object has hardly been performed in the existing research. Therefore the contribution of this work is to cover the evaluation of end-user strategies for programming everyday objects.

			End-User Programming						
System name	Application area	End-User programming	GUI	PBD	Speech	AR	Technology	Use of affordances	Evaluation performed
Worldkit	instant control: interactive sufraces	no					Depth camera, projector	yes	no
Opportunistic Music	instant control: music	yes	x	x			2 cameras for identifying 3D position, FSR sensor	yes	no
OnObject	instant control: learning languages, storytelling	yes		x			RFID, accelerometer	yes	yes (interaction)
Building Upon Everyday Play	instant control: gaming	yes		x			Based on Exemplar, motion sensing and pattern recognition	yes	yes (interaction)
iCon	instant control for everyday tasks	yes	x				Fiducials, vision based	yes	yes (interaction)
Smarter Objects		yes				x	AR	yes	no
Instant User Interfaces		yes	x	x	x		Marker-free tracking, Kinect	yes	yes (programming strategies)
Jabberstamp	instant control: drawing	yes		x			Wacom tablet, speakers, microphone	no	yes (interaction)
IOBrush		yes		x			Video camera, touch and light sensors, Wacom screen with a tablet	no	yes (interaction)

Figure 2.12: Summary of the related work: systems that enable everyday objects to be used as controllers.

2.2 Programming by Demonstration

In this section we look into the *Programming by Demonstration* in more detail and examine in which application domains and for which tasks it has been used.

Cypher et al. [1993] described how users could create programs without learning a programming language, but by instructing a computer to "watch what I do". According to Cypher the idea behind Programming by Demonstration is that *"if the user knows how to perform a task on the computer, that should be sufficient to create a program to perform this task"*. For instance, a computer could remember a sequence of user's actions in a program and automatically repeat it. Since 1993 this idea has been deve-loped further and applied in different application domains, such as robotics and rapid prototyping of interactive user interfaces. Terminology of PBD varies depending on the application domain, we formulated we following generic definition of PBD based on the related literature:

Definition: *Programming by Demonstration*

PROGRAMMING BY DEMONSTRATION:
end-user technique for teaching a system to perform in a certain way by demonstrating the behavior without wring code in a programming language.

Further in this chapter we will look at the examples of projects using PBD in different application domains and summarize our findings about the benefits and motivation of PBD approach in Section 2.2.4 "Summary: Programming by Demonstration".

2.2.1 User-Centered Programming by Demonstration - Stylistic Elements of Behavior: Young et al., 2013

Programming by Demonstration has been commonly applied in robotics, for instance, for programming natural human-like movements. Young et al. [2013] presented a project for modeling robotic locomotion styles: e.g., a robot

can move aggressively or politely in response to certain actions from the person. In the project robot's behavior could be implemented in two ways: via writing Java code based on the provided API, and by demonstration. Authors conducted a user study to compare these approaches. Participants were experienced programmers and each of them was asked to program the robot's locomotion style by writing code and by demonstration.

PBD has been used in robotics for mimicking human-like actions and movements.

For programming the robot's motion by demonstration users were asked to move it with a designed for the task broomstick (Figure 2.13). This way the movement could later be directly applied to the same kind of robot as it encapsulates the robot's movement properties. After the robot's motion style has been programmed, the robot could move independently and follow a person in a polite way or aggressively (following a person was not a part of demonstration, but it was provided via a special algorithm).

A robot could be trained by PBD to follow a person politely or aggressively.

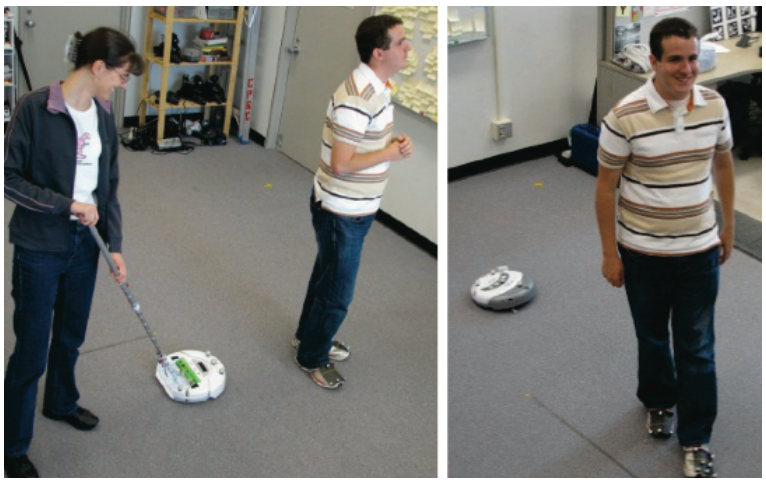


Figure 2.13: Programming a robot's behavior by demonstration

During the study users were given two hours for programming a motion by code, and the time for the demonstrational approach was not strictly limited. As a result users took full two hours for completing programming the stylized behavior in Java, and it took them 15-30 min to create robot's movements by demonstration. Users expressed that the broomstick interaction was easier than programming. However, programmers also mentioned that they felt li-

A user study compared two ways of programming robot's movements: by writing Java code and by demonstration.

mited by the system, for instance, they could not convey to the robot to stay away from the corners. Experimenters observed that users acted the characters (with body motion, making faces, etc.) according to the behaviors they were conveying to the robot. Young et al. [2013] noted that demonstration "makes sense to people and involves their innate social and emotional interaction skills", and these benefits of the PBD approach will appear stronger to non-programmers, as to users who do not have an alternative to program the interaction in code.

Benefits of the PBD might appear stronger to non-programmers.

Users did not show clear preference, but pointed out some trade-offs between the approaches. For instance, several programmers expressed that by writing code they felt more in-control, as they had less limitations from the system. However, writing code and its debugging is time-consuming and PBD can be a good alternative for creating medium fidelity prototypes.

Writing code let programmers feel more in-control and not dependent on the system's restrictions.

- Young et al. [2013] suggest that PBD adds a complementary user-centered contribution to the stylistic programming of robotic behavior.
- PBD can be used for creating medium fidelity prototypes, as this approach is significantly faster than writing and debugging code.

2.2.2 Exemplar: Hartmann et al., 2007

In a system called *Exemplar* Hartmann [2007] introduced a set of techniques for creating sensor-based interactions by demonstration. Exemplar makes the process of creating rapid prototypes with sensor-based data accessible to designers and programmers without deep knowledge of pattern recognition. Making the process of rapid prototyping accessible to a wider range of people encourages exploration of new interactions. Authors emphasize the value of "epistemic experience of exploring alternatives" during the process of creating interactive systems.

PBD has been applied in creating sensor-based prototypes.

Exemplar system allowed to map sensor-based input to application controllers.

Designers begin prototyping with connecting hardware sensors to the PC running Exemplar software. Then a user (designer is the typical end-user of the Exemplar system) decides which sensor he would like to use and activates it (e.g., a force sensor). The system recognizes the input from the sensor and visualizes it, the user then can set the threshold or filter for recognizing this particular interaction. For connecting sensor output to the application, Exemplar converts the data streams into operating system input events such as key press or mouse movement (Figure 2.14). The output can be mapped into continuous or discrete event.

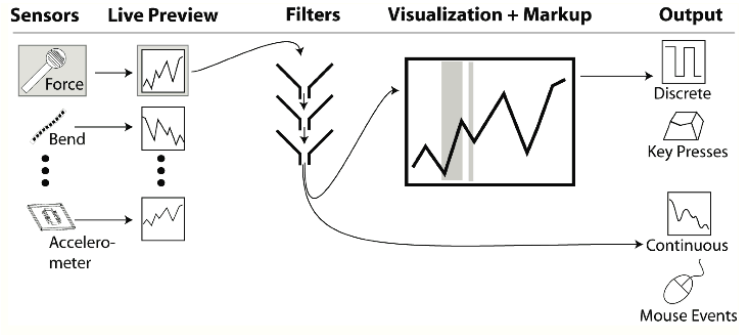


Figure 2.14: Exemplar system recognizes the input from the sensor and visualizes it, a user then can set the threshold or filter for recognizing this particular interaction and map it to a continuous or discrete event.

Evaluation of the *Exemplar* system and PBD approach included applying a *CDN* framework and conducting a lab study.

Hartmann [2007] performed an extensive evaluation of the system and the implemented PBD approach: first, the authors applied the *Cognitive Dimensions of Notation (CDN)* framework by Green and Petre [1996] to evaluate the design trade-offs of the Exemplar system, and then the researchers conducted a lab study to assess system's usability. In summary, Exemplar performed good on visibility (all current sensor inputs are visible to the user), closeness of mapping (direct presentation of the signal appeared to be consistent with the user's mental model), and progressive evaluation (real-time feedback was provided by the system and users could save the current state of a session to continue later). More details of the CDN evaluation are presented in the paper by Hartmann [2007].

PBD motivated users to experiment and it allowed designers to better understand user experience with sensor-based input.

The results of the post-experiment questionnaire are shown on Figure 2.15. For instance, users agreed that Exemplar decreases the time needed for creating prototypes with sensor-based input and that the system motivated and let users experiment more. Another interesting finding, as expressed by users, is that Exemplar allowed to better understand user experience, since designers could try out the sensors during the demonstration.

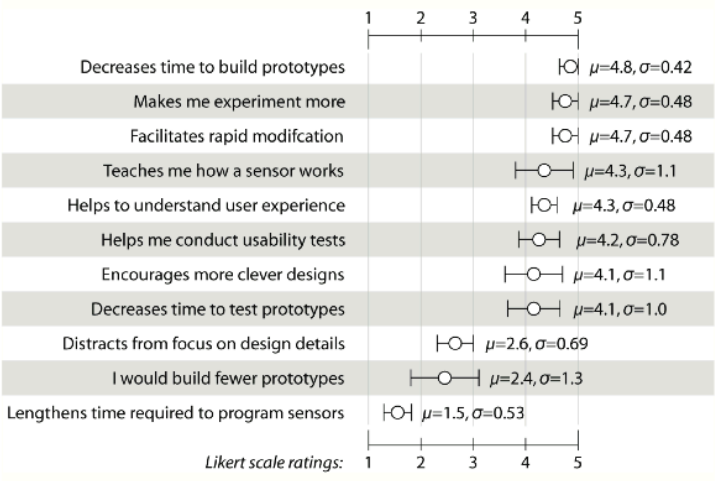


Figure 2.15: Exemplar: evaluation results show that according to users' feedback Exemplar decreases the time needed for creating prototypes with sensor-based input and that the system motivated and let users experiment more

- Hartmann [2007] showed how PBD can be used in rapid prototyping for systems using sensor data. The implemented programming strategy enabled wider range of users (designers, programmers without knowledge of pattern recognition and other participants of the design process) to create interactive prototypes .

- Hartmann [2007] performed an extensive evaluation of the Exemplar system, that showed that users were able to create rapid prototypes faster, and that they could better understand the user experience, as users got to experiment with the sensors and try out different interactions during demonstration.

- PBD lowers the threshold of technical knowledge needed to create interactions, ans our vision is that PBD could be a suitable strategy for programming everyday objects by *end users*.

2.2.3 Touch and Activate: Ono et al., 2013

Ono et al. [2013] presented an acoustic touch-sensing technique called *Touch and Activate*. The setup consists of a vibration speaker and a piezo-electric microphone, which are attached to an object or a surface. The main principle is based on the fact that each object has its resonant frequency, and it changes when the object is touched. By measuring the resonant frequency, the system can distinguish between different ways the object is touched or held.

Touch and Activate presented a technique to sense touch applied to different areas on an objects by measuring the resonance frequency.
A controller could be mapped to a touch applied to an area on an object by demonstration.

Authors developed several applications to demonstrate possible use scenarios of the *Touch and Activate* technique. For instance, they developed a music player that can be controlled by touching different areas on an everyday object. In order to map an area on the object to a controller, a user would need to train the system *by demonstration*: he would touch the object and press the key on the keyboard associated with a controller. Figure 2.16 shows a user mapping an area on a LEGO block to a *Play* controller by demonstration.

The evaluation performed by Ono et al. focused on measuring the classification accuracy. During the study participants were shown the pictures of gestures for 7 seconds and asked to perform those gestures (while continuously adjusting th gesture posture). The results of the study are presented in the paper by Ono et al. [2013] and showed a good accuracy that proved that *Touch and Activate* can be used for rapid prototyping to enable touch input capability.

Machine learning methods can be applied to increase the benefits of PBD: a system can be trained to recognize specific touch applied to an object.

- Ono et al. [2013] presented a touch-sensing technique that allows to map the ways an object is touched or held to controllers of technical systems. Grippo also provides the functionality of mapping different grip

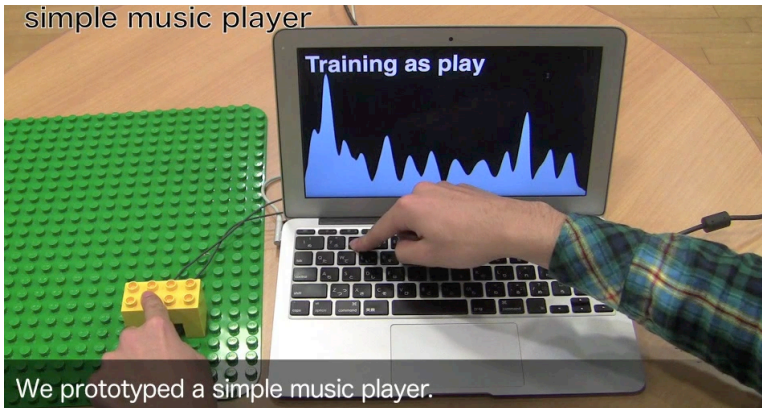


Figure 2.16: Mapping an area on the LEGO block to the *Play* controller: training the system by demonstration is done by touching the area on the object and pressing the key on the keyboard associated with a controller.

gestures to digital controllers. Although we used a vision-based approach for tracking the objects (that is beneficial for tracking how the object has been moved), the *Touch and Activate* technique could be used as a complementary implementation approach for the object's touch and grasp recognition.

- In *Touch and Activate* the demonstration is part of *training* the system to recognize a touch gesture. This is yet another application for PBD as it allows to teach the system to recognize new interactions and thus make the system more flexible and personalized.

2.2.4 Summary: Programming by Demonstration

PBD is often applied in systems using sensor-data, motions and touch.

A short overview of systems using PBD gave us some interesting insights. We saw that PBD is often applied in the systems using sensor data for recognizing gestures, motions and touch, as in these systems the user *demonstrates* certain action that can later be mapped to a controller. Modern systems not only record user's actions in the GUI, but also can understand gestures and tangible interactions.

PBD makes technology more accessible on different levels: it enables developers and designers to improve prototyping process of interactive systems, by making it faster and simpler (e.g., *Exemplar* system by Hartmann [2007]), but it can also give end users, who don't have any technical knowledge, the power of creating new interactions and personalize the application.

PBD makes prototyping interactive systems using sensor and motion data accessible to a wider range of users.

Based on the analysis of systems using PBD, we created a list of motivations of a PBD approach:

- Automation of repetitive tasks

  Automation of repetitive tasks is one of the conventional applications of PBD. We did not describe such systems in this section, since it is out of our scope, but one of the example is a mobile application called *Keep Doing It* by Maués and Barbosa [2013]. The application continuously records user's actions and then lets him to create an automation based on the latest actions, e.g., when the wired headset is connected start the music application. This automation of tasks can make the smartphone easier to use and more battery efficient [Maués and Barbosa [2013]].

- Lowering the threshold of technical knowledge

  PBD enables users to program certain tasks with little or none programming knowledge. It encourages a wider range of users to experiment with interactive interfaces and motivates creativity. In Grippo we use this characteristic of PBD to enable end users to program everyday objects with grip gestures.

- Improving the process of prototyping gestural and tangible interfaces

  As PBD lowers the threshold of technical knowledge needed to create interactive systems, it improves the process of prototyping by making it simpler and faster, thus allowing more design iterations and improving end-user experience.

- User-centered programming approach

  In robotics PBD is considered a user-centered programming approach, as it implies user-centered requirement, e.g., stylistic behavior that is clear to people [Young et al. [2013]].

- Personalization

  PBD empowers end users to personalize their interactions with technical systems, moreover, it allows to create new personalized interactions by training the system with AI techniques [Ono et al. [2013]].

PBD can be a promising approach for programming everyday objects.

We consider PBD a promising approach as an end-user strategy for programming everyday objects. It can let users personalize their interactions without a need of programming or any technical knowledge. With Grippo we will evaluate PBD for repurposing everyday objects as controllers and compare it with another common end-user programming strategy – a GUI-based approach.

2.3 Grasping Gestures

In this section we will look at the research related to grasping gestures. We will examine classifications of hand grasp gestures and see how implicit information conveyed through a grasp can be used in designing graspable interfaces.

2.3.1 The Prehensile Movements of Human Hand: Napier, 1956

[Napier [1956]] conducted research of *prehensile movements* of human hands.

Definition: *Prehensile movements*

PREHENSILE MOVEMENTS:
movements in which an object is seized and held partly or wholly within the compass of the hand.

Napier analyzed prehensile movements of human hand from anatomical and functional viewpoints. One of the results of his analysis was the identification of two basic patterns in prehensile hand movements: *precision grip* and *power grip*.

POWER GRIP:
a grip when an object is held in a clamp formed by the partly flexed fingers and the palm, counter pressure being applied by the thumb lying in the plain of the palm.

Definition: *Power grip*

PRECISION GRIP:
grip during which an object is pinched between the flexor aspects of the fingers and the opposing thumb.

Definition: *Precision grip*

Figure 2.17 demonstrates a person unscrewing a lid of a jar. In one case, when the lid is tightly screwed-up, a person uses power grip, that allows to exert more power, but as the lid loosens, a person changes the way he is holding an object to a precision grip.

Power grip allows to apply more power in an action applied to an object compared to the precision grasp..

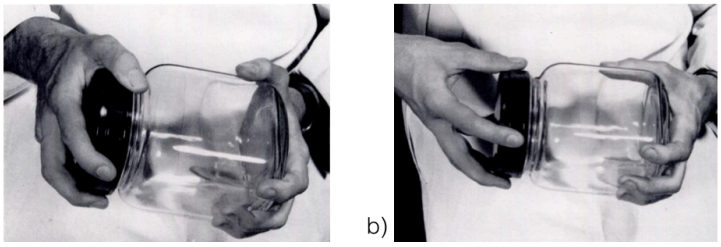


Figure 2.17: a) A person uses power grip to unscrew a tightly screwed-up lid of a jar, b) a person changes the grip to a precision grip as the lid loosens.

Napier shows that, as an activity influences the grip, then *"the nature of a prehensile activity can be resolved into two concepts – that of precision and that of power"*. During the development of Grippo we considered using this concept. For instance, if one task has two degrees of how it can be performed, such as fine and coarse video navigation, then precision grip can be applied to an object (e.g., water bottle) for slight adjustments (fine navigation) and power grip could be used for skipping multiple frames (coarse naviga-

The concept of power and precision grips can be applied in using everyday objects as controllers if the controlled property has several degrees of how it can be applied.

tion). Similar metaphor could be applied in zoom functionality for enhancing an image or a map with different speed. These tasks were out of scope of current work, however, they can be implemented in future as an extension to the developed system.

2.3.2 Grasp Sensing for Human-Computer Interaction: Wimmer, 2011

User's intentions can be implicitly conveyed through a way he performs a grasp.

Grasping gestures often convey information about the goal of the person performing it. The extraction of this information, according to Wimmer [2011], can be used for improving interaction with graspable user interfaces. Wimmer presented a GRASP framework where he identified five general factors that determine a grasp: *goal, relationship, anatomy, setting* and *properties* (Figure 2.18).

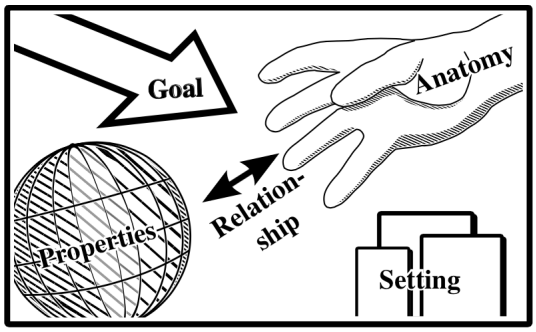


Figure 2.18: The GRASP Model includes five factors influencing a grasp: goal, relationship, anatomy, setting and properties of an object.

According to the GRASP model there are five main factors that describe a grasp gesture.

- **Goal** Goal is an important factor of defining a grasp. When people perform an *implicit* grasp (meaning that its goal is to *manipulate* an object), it may carry information about user's intentions. It may be a primary grasp aiming to move an object or a supportive grasp for fixating a position of the object for further interaction. Extracting this information can be used in building user interfaces: for instance, if a person is putting a mobile phone to his ear (primary grasp), this could automatically trigger answering the

call. *Explicit* grasps, that are performed to trigger an effect, always carry information about person's goal and should be treated as meaningful. An example of explicit grasp could be a user authenticating himself to a device by grasping it in a certain way.

- **Relationship**

  Relationship between the person and the object can also influence the grasp. For instance, such feelings as fear or disgust can define how the person is holding or touching an object.

- **Anatomy**

  The size of person's palm, as well as the length and number of fingers can lead to different grasps. Moreover, people can apply different force in grasping an object.

- **Setting**

  The environment and setting also influence the grasp. For instance, grasps will be different if a person is trying to reach for an object or to pull it from somewhere. Other factors in this category include lightning conditions, available space and temperature.

- **Properties**

  Properties of an object can greatly influence the grasp. Object's size, shape, texture or weight can define how a person applies a grasp.

GRASP model can be applied to the Grippo system.

If we apply the GRASP model in Grippo, we can see that the *properties* and the *setting* were constant. The setting of the study was fixed as well as the two objects we chose for our prototype system. The *anatomy* factor varied to some degree, as everybody's hand anatomy is different, however, in case of our study, the recognizable grasp gestures were simple and accessible to everybody. The *relationship* factor can be ignored in our example, since users had a neutral relationship to the presented objects. The *goal* is the most important factor that influenced how a person performed a grasp. Users could assign certain grasps to an object for using it as a controller, thus giving those grasps a new meaning, while other ways of grasping an object would not trig-

Extracting user's goal from an applied to an object grip gesture may allow distinguishing between the normal and repurposed use of an object.

ger any action. This can help to distinguish between the normal use of an object (e.g., in case of a bottle – pouring water to a glass) and the repurposed use (e.g., using an object as a volume controller).

2.3.3 Summary: Grasping Gestures

Research has been performed in classification of grasp gestures and has resulted in different taxonomies. We described a classification of prehensile hand movements into *power grip* and *precision grip* by Napier [1956]. The advantage of this classification is its general application independent from other factors and activities. The concept presented by Napier can also be applied in the domain of using everyday objects as controllers: for instance, power grip can be associated with activities where users *"exert more power"*, such as coarse video navigation or fast zoom, in comparison to fine navigation and more detailed zoom, where a precision grip applied to an object could be more appropriate.

Research in grasping gestures can be used in HCI for improving user experience with graspable interfaces.

Moreover, grasping gestures often convey implicit information about the goal of the person performing the grasp, his relationship to an object and other facts. Extracting this information from a grasp gesture can improve graspable interfaces. In the domain of using everyday objects as controllers this information can help to distinguish between the normal and repurposed use of an object.

Chapter 3

Preparation Phase

3.1 Brainstorming

End-user strategies for programming everyday objects have been used but have hardly been evaluated in existing research.

Literature research has shown that there are different approaches for end-user programming of everyday objects: *Programming by Demonstration*, using *Graphical User Interface*, *Augmented Reality approach* and *Speech control*. Objects can also be activated by *pointing* and combinations of these approaches are possible: e.g., activating an object by pointing it at the GUI interface [Corsten et al. [2013]]. However, these end-user programming methods have hardly been evaluated. Most of the research in the field of using everyday objects as controllers focuses on the *interaction with an object as a controller* and not on *programming an object*. Therefore, the goal of this thesis is to examine and evaluate the end-user programming part.

Application domain for the prototype systems

We chose interactions in the domestic environment for our sample scenarios, as it demonstrates how using everyday objects as controllers can serve *convenience*.

In order to define the programming strategies for evaluation and comparison, we first, needed to define the tasks – what would users want to program, what kind of functionality would they want to assign to the object. One of the common application domains for using everyday objects as controllers is home automation. It is the environment where users generally have physical objects suitable for appropriation at reach (objects should not be too heavy

or fragile, or expensive [Cheng et al. [2010]]). Moreover, in a domestic environment, users have different electronic devices, that need to be controlled remotely (e.g., TV, Lights, Music). *Convenience* of controlling devices remotely is one of the reasons why users could repurpose everyday objects. Other reasons include: *unavailability of the dedicated controller* (broken remote control), *spontaneous access* to the objects (everyday objects could be used as they are, without complex technical preparation), and *vicinity* of everyday objects. Moreover, *physical affordances* of everyday objects can be exploited to provide *eyes-free interaction and haptic feedback*.

Programming everyday objects using grip gestures

We added grip gestures to programming everyday objects for reusing one object for multiple controllers.

In addition to repurposing an object as a controller, we decided to let users assign multiple controllers to an object using different *grip gestures*. The motivation for this decision was based on the fact that typical controlling devices provide multiple functions, for instance, a TV remote control allows to switch the device on and off, adjust volume and screen brightness, navigate between channels, etc. With different grip gestures users can assign multiple controllers to one object similarly to the dedicated controlling devices. For instance, if a user wants to temporary substitute the functionality of a TV remote control, he will not need to use a separate object for each function, but he will be able to reuse the same object for several functions instead.

Choice of the end-user programming strategies for implementation

Once we have decided what tasks we will implement and that the designed system will enable users to assign controllers to an object using grip gestures, we needed to choose end-user programming strategies for further implementation.

Our vision is that PBD supports the idea of *Instant User Interfaces*.

Initially our vision was that *programming by demonstration* supports the idea of Instant User Interfaces [Corsten et al. [2013]], as it suggests direct interaction with object. However, implementations of PBD may vary. For instance, a *"direct"* approach, not requiring additional devices, would

be to connect an object with a controller by a simultaneous touch (e.g., tap on top of a bottle and touch the TV power button to create a mapping). A context-aware approach, that could enable such interaction, was presented by Park et al. [2006]. This approach, called *Touch-And-Play (TAP)*, allowed users to e.g., connect a camera with a TV to start a slide show: if a user holds a camera in his hand and touches a TV, data communication could be enabled trough a human body. One of the drawbacks of this approach is that the vicinity of a user holding an object and a device (controller) is a requirement for such interaction. To overcome this drawback, we could apply *pointing* at the target device instead of touching it. However, if a user points at a device, e.g., a TV – it is not clear which TV function he wants to map (Figure 3.1).

Implementations of PBD may vary and PBD can be combined with other programming strategies, such as *pointing*.

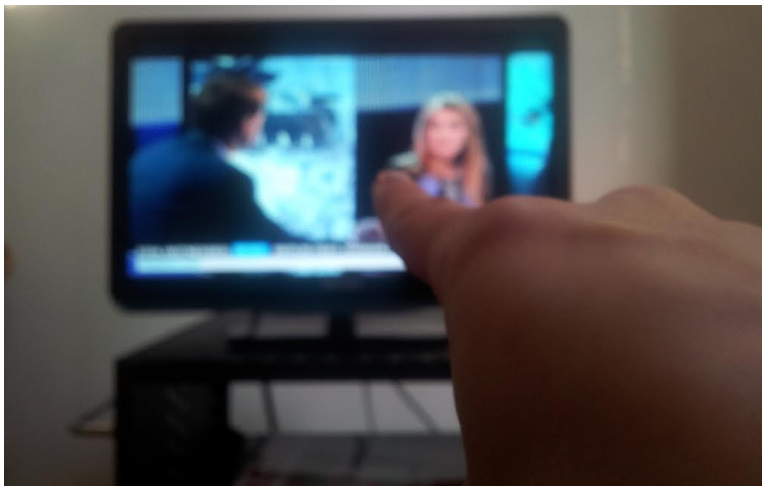


Figure 3.1: If a user simply points at a device, without invoking an additional interface, it is not clear which functionality he wants to map to an object – in case of a TV it could be power, volume, navigation between channels, etc.

Speech control can also be used as means for programming everyday objects. Although, one of the known drawbacks of speech control is its intrusiveness, it could be used in combination with other programming approaches, such as demonstration. Further investigation of speech control as an end-user approach for programming everyday objects is out of scope of this work.

We implemented a PBD approach combined with a smartphone application.

Because of the described limitations of the *"direct demonstration"* without intermediate device and the *pointing* approach, we decided to implement a hybrid PBD approach complemented with a smartphone. This combined approach provides users an instrument for selecting controllers and managing created mappings: with a smartphone application users can map new controllers to an object, change and delete mappings and have a reference of currently assigned controllers. The smartphone will be used only for programming an object and not during the interaction with an object as a controller.

GUI-based strategy was chosen as a common approach for programming objects for comparison against the PBD.

We also wanted to compare the PBD approach with another common strategy for programming everyday objects – *a GUI-based approach*. Since we decided to use a mobile application in the PBD prototype, we will also use a smartphone in our implementation of the GUI programming strategy.

Types of controls for the implementation

Before implementing actual tasks from the chosen application domain, we defined the types of controls we wanted to focus on. Based on the relevant real world tasks (i.e., domestic interactions in a living room), the following types of controls could be implemented:

- Binary or discrete controls

  Examples of binary control are push buttons changing a property either between two states, e.g., power on and off, play and pause, mute and unmute, etc., and discrete controls allow switching between multiple states, e.g., going to a next channel, increasing volume by pushing a button, etc.

- Continuous controls

  Continuous controls are associated with properties, controlled by a rotary knob, for instance, changing a radio frequency, dimming the lights, or adjusting ventilation in a car.

- Sliders

  Sliders could be seen as a subcategory of continuous controllers, however, if we look at video navigation

bar as an example, the difference is that in this case the slider is changing its state independently from the user as the movie is playing.

There are tasks that could be accomplished at the same time by each of these types of controls, e.g., volume can be adjusted by pushing a button, rotating a volume knob or by moving a slider. We divided tasks in these categories to better understand how to map it to everyday objects.

We include binary, discrete and continuous types of controls in the implementation of software prototypes.

Since the goal of this thesis is to investigate end-user programming strategies, we decided to choose a representative, but limited set of interactions. We will implement sample tasks of a binary control (e.g., TV power, light switch, etc.), a discrete control (e.g., navigation to the next and previous TV channels) and a continuous control (e.g. TV volume, lights brightness, etc.). We will leave out repurposing an object as a slider for video navigation, since it cannot be actuated as the movie is playing, this can lead to inconsistent states. In future work, video navigation with an everyday object can be implemented by repurposing an object as a rotary knob.

3.2 Prototyping

Several iterations of paper prototypes and interactive mock-ups were done before finalizing design decisions about software prototypes.

Before implementing software prototypes we went through several iterations of paper prototypes and interactive mock-ups. Further in this section we describe some of our findings during the prototyping phase.

First interactive prototypes were created with MS PowerPoint presentation software. We created mock-ups for the PBD and GUI-based applications. In the first iteration we used a mug as an everyday object, that was repurposed as a volume controller. In both applications a programming sequence started from the choice of a target device and its controller and was followed by the selection of an object and a grip gesture. Then in the initial prototype a user would define object positions for minimum and maximum values. In this case, a handle of a mug could serve as a state

First iterations of interactive prototypes included a step of defining object positions for the minimum and maximum values of a property.

indicator. Figure 3.2 illustrates a process of mapping object's positions to the minimum and maximum values. In the GUI-based prototype a user could manipulate a virtual representation of an object on the screen, and in the PBD prototype a user could put the real object in the position that he wanted to be associated with a respective value and capture it with the smartphone camera.

Figure 3.2: Mapping the positions of the mug to the minimum and maximum volume values: a) with a GUI-based approach by manipulating a virtual object on the screen, b) with the PBD approach by setting the real mug in a desired state in front of the smartphone camera.

In further iterations of prototyping we decided to skip the step of defining positions for minimum and maximum values, and use objects, that do not have possible state indicators.

After an expert evaluation of these prototypes we decided to skip the step where users define minimum and maximum values of a property, as it was making a programming sequence longer and, more importantly, it could result in state inconsistencies if the property (e.g. volume) was adjusted from the original device. Moreover, in the follow up prototypes we decided not to use objects with possible state indicators, such as a mug with a handle. Instead, we chose to use objects such as a glass or a bottle, that do not have such indicators. It also helps to achieve state consistency – even if the property was adjusted from the original device, since there is no state indication – the current value of the property can be sent to the object position and then adjusted by the user relatively to this state.

In the next prototype iteration we used a glass as a sample object. Figure 3.3 illustrates a part of the PBD sequence: a user selects an object and then demonstrates the grasp

gesture he would like to map to the chosen controller.

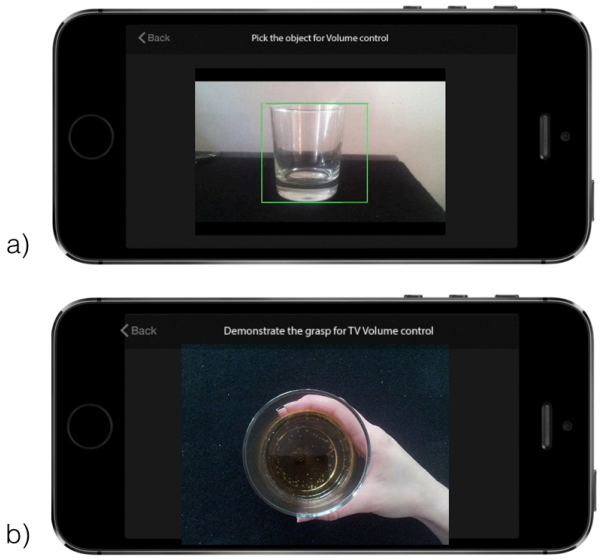


Figure 3.3: Using an object without visible state indicators, such as a glass: a) a user, first, selects an object, and then, b) demonstrates the grasp we would like to map to the selected controller without indicating object positions for the minimum and maximum values.

We noticed that in this prototype the information about currently assigned mappings is missing, i.e., when a user selects an object, he does not know which controllers and grip gestures have been assigned to it. This functionality is important and it will be implemented in the later stages of prototyping, the final version of the sequence of actions is described in Section 5.3 "User Action Sequences".

Information about currently assigned mappings should be available to the user.

Prototyping allowed as to identify major flaws in the interaction and define the general sequence of actions for the software implementation. The next step was to identify which grip gestures should be implemented, select the technology for tracking these grip gestures and start programming the final software prototypes.

Chapter 4

Preliminary Study: a Survey on Grasping Objects of a Cylinder Shape

4.1 Description

With the preliminary study we wanted to identify users' preferences in grasping objects of a cylinder shape.

The goal of the survey study was to get preliminary information about grip gestures preferred by users. The results of this preliminary study provided input for the implementation of software prototypes. The study was designed to answer the following research questions:

- Which grasps of the objects of cylinder shape are preferred by users?
- In case of assigning multiple grip gestures and controllers to the object which combinations of grip gestures are preferred? Do users prefer using different areas of the object or different number of fingers in a grip to assign different controllers to an object?
- For a binary control such as a push button — can a tap on a firm surface (without an affordance to be pushed) be used as a binary control?

- Do users prefer to reuse an object and map several controllers to it or take a separate object?

4.2 Procedure

Although we conducted the survey online, we asked users to interact with a real physical object (a glass) from their environment.

In total 14 users aged 22-30 (M=24, SD = 2.46, three females) took part in the survey. Users were presented an online questionnaire, that consisted of six tasks. In each task users were asked to select a grip gesture or a combination of grip gestures applied to a glass from the list of options. The order of options in each question was randomized. Before answering a question users were asked to take a glass at home at try to grasp it in different ways. Although the provided set of grip gestures and their combinations was limited, users were given a possibility of suggesting their own answer. Full questionnaire can be found in the Appendix A "Appendix for the Preliminary Study: a Survey on Grasping Objects of a Cylinder Shape".

4.3 Results

Users preferred to use the *whole hand* and hold an object from the side or apply a *a four finger grasp* from the top of the object for grasping and rotating a glass.

In the first question users were asked to take an object and try to use it as a rotary knob. Results are presented on the Figure 4.1 and show that the most common ways to grasp an object of a cylinder shape (e.g. a glass) and rotate it are with the *whole hand* from the side (five users) or with *a four finger grasp* from the top of the object (four users). The answers of the rest five users were spread between other options.

In the next question we added another controller to the task and asked users to select or suggest a combination of grip gestures. As a result, eight participants chose a combination of a top grasp with the whole hand and a side grasp with a whole hand (Figure 4.2). This means, as we expected, that users preferred to assign different controllers to *different areas on the object* rather than to use different number of fingers for different tasks. A possible explanation

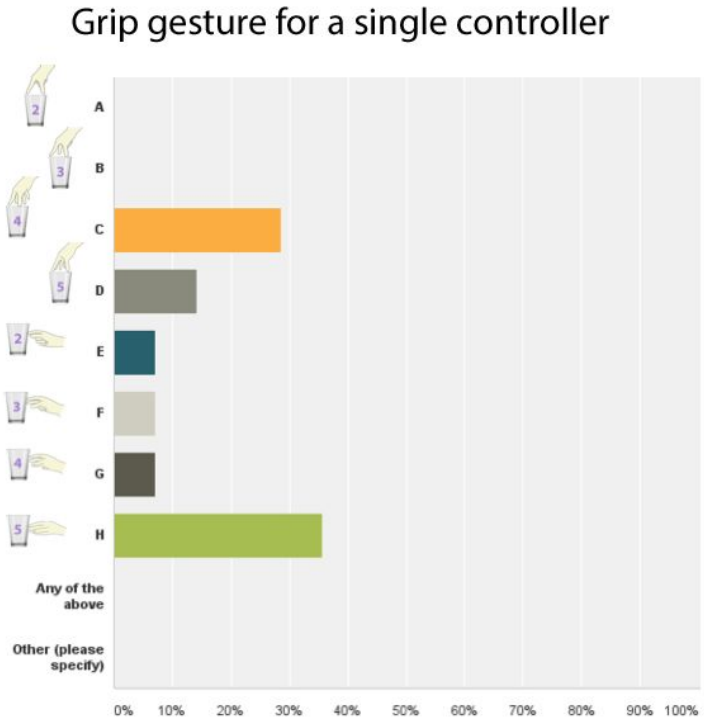


Figure 4.1: Users' choices of a grip gesture for a single controller (the number on the glass is the number of fingers in a grip). Most common ways to grasp an object of a cylinder shape appear to be with the whole hand from the side or with a four finger grasp from the top of the object.

could be that different areas are easier to remember compared to combinations of fingers.

Users preferred to use different areas of an object compared to using different number of fingers in a grip when using an object for multiple controllers.

When users were asked to assign three controllers and select a combination of three grip gestures to the glass, we saw again that using different number of fingers for different tasks was the least preferable option (only chosen by one participant). The most common choice (by eight users) was using a top grasp and two side areas on the object (Figure 4.3). Two participants selected an option of three different areas on the side of the object, that also supports the idea that users prefer mapping areas compared to combinations of fingers in a grip gesture. However, another interesting finding was expressed by three other users: instead of using areas on the object they suggested different

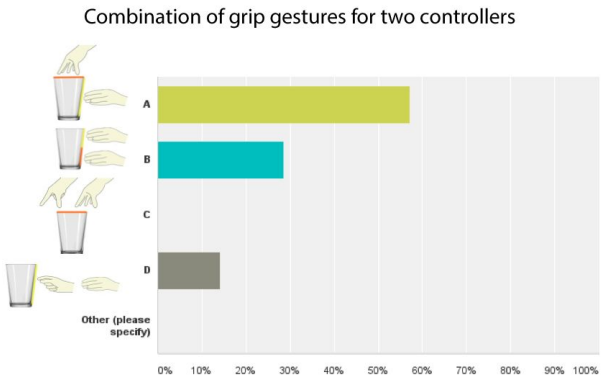


Figure 4.2: Users' choices of grip gestures for two controllers show that it is preferable to grip an object on different areas compared to using different number of fingers in a grip.

Users suggested performing different *actions* with an object for distinguishing between the controllers.

actions. For instance, use an object as a rotary knob for one controller, use it as a slider for another one and slide a finger along the top edge of the glass for the third controller. Another suggestion was to include the bottom surface of the glass while holding the object in one hand and slide the finger clockwise along the bottom edge to increase a value and slide the finger counterclockwise to decrease it.

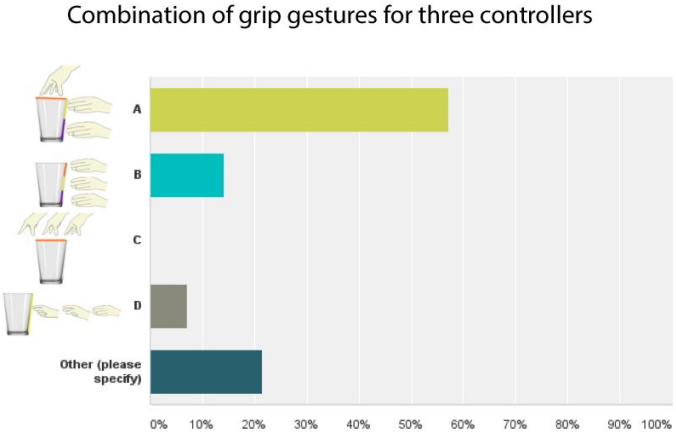


Figure 4.3: Users prefer to distinguish between different areas of an object compared to using a different number of fingers in a grip. Moreover, if an object is to small for defining three clear areas for grips, different actions can be performed with an object for representing different controllers.

In the next question we asked participants to select a combination of grip gestures for controlling two degrees of one function, the suggested example was fine and coarse video navigation. Our vision was that for this kind of task a user would not like to move his hand a lot and instead he would use e.g. a grip gesture with two fingers for fine navigation and with five fingers for coarse one using the same area on the object. However, the results were controversial and we did not observe any clear trends for this question (Figure 4.4).

Grip gestures for controlling degrees of one function need to be investigated further.

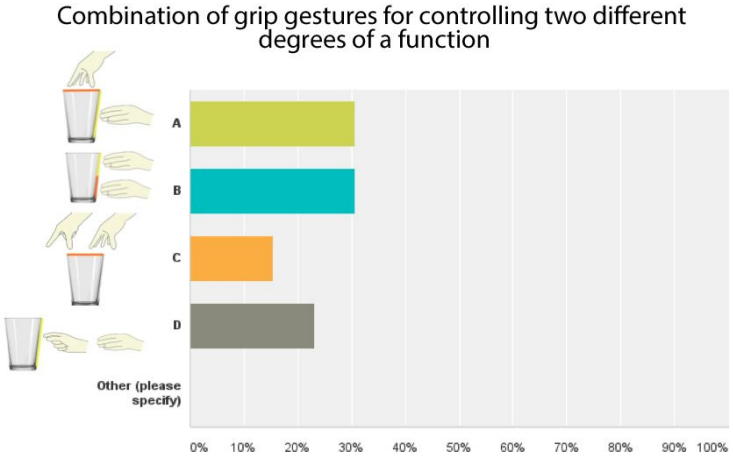


Figure 4.4: Users' choices of grip gestures for controlling two degrees of one function did not show clear preference and need to be investigated further.

When users were asked to assign a binary controller such as switching a device on and off they were given two choices: to use a tap on top of the glass or to use an object with a "push" affordance, such as a stapler. Results (Figure 4.5) show that seven users chose to push the stapler, six users chose a tap on top of the glass (and one user did not answer this question). Although, due to the respective affordance, an object that can be pushed, could have been a more popular choice, there was no preference for assigning binary control to an object between the suggested options.

Participants did not show clear preference in using objects with or without a push affordance for mapping a binary controller.

In the last question we described a situation when one controller (e.g. TV volume) has already been assigned to the

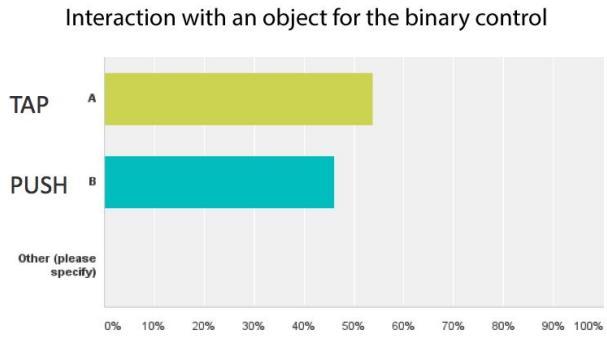


Figure 4.5: Participants did not show a preference as a group for using an everyday object with or without a push affordance for a binary controller.

Participants preferred to reuse an object for multiple controllers.

object and we asked participants to assign another binary control (TV power). Participants could either reuse the object and add a tap on top of the glass or they could choose to take another object (e.g. stapler) and map new controller to it. Most of the participants (10) preferred to reuse the object and three participants chose to take a new one (Figure 4.6).

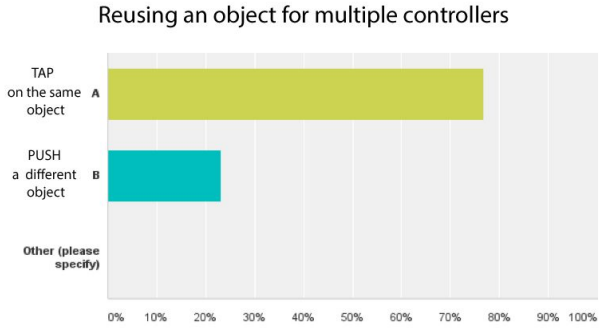


Figure 4.6: Ten participants preferred to reuse an object for multiple controllers and three participants chose to take a new object with a push affordance.

4.4 Findings

As a result of this preliminary study we saw which grip gestures of objects of a cylinder shape are preferred by

users. We found that users prefer to map multiple controllers to different *areas* of the object compared to using different *combinations of fingers in a grip gesture*. We will use this finding in the software implementation and track when users grip an object in different areas (the description of implemented grip recognition is presented in Section 5.2 “Implemented Grip Gestures”)

Results of the preliminary study provided input for design decisions about implementations of grip gestures.

Users did not show clear preference between assigning a binary controller to an object *with* or *without* a clear ”push” affordance.

We also found that it is reasonable to use a tap on top of the surface without push affordance, especially in cases when the object is reused for multiple controllers. Therefore we will implement a tap on top of the bottle as a controller and tap on different areas of a chocolate bar for mapping several binary or discrete controllers (the list of touch-sensing areas of the objects is presented in Section 5.2 “Implemented Grip Gestures”).

Chapter 5

Software Prototypes

We developed two software prototypes representing a GUI-based and PBD strategies for programming everyday objects. In this chapter we explain how the tracking of objects and grip gestures was done, and describe implemented applications in terms of user action sequences. Then we cover overall system architecture and describe the data communication between the interoperating parts.

5.1 Tracking Technology

We used Vicon system[1] for tracking grip gestures performed by users and position of the objects. The tracking system consists of eight Vicon Bonita cameras emitting infrared (IR) light, a set of reflective markers and Vicon Nexus software. The system provides millimeter accuracy and is appropriate for rapid prototyping: in order to start working with Vicon, the system needs to be calibrated, then if a marker is visible for at least two cameras, it can already be tracked by the software. Figure 5.1 shows the objects augmented with reflective markers and the perspective view in Vicon Nexus system.

Vicon Nexus system was used for tracking grip gestures and positions of the objects

[1] http://www.vicon.com

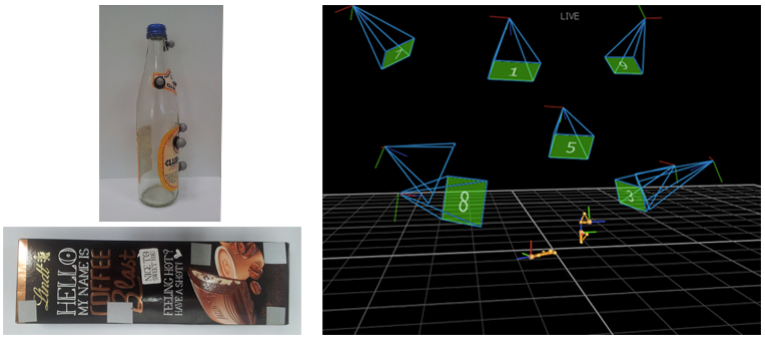


Figure 5.1: Two objects selected for the user study were augmented with reflective markers. When a marker is seen by at least two cameras, it can be tracked by the Vicon Nexus software.

However, for stable tracking, the markers have to be visible to the system at any time, and when people manipulate objects with markers attached to it, they might accidentally cover some parts, thus affecting the tracking. This is a limitation of the described tracking approach and for this reason, the set of the grip gestures that were recognized by our prototypes is restricted.

5.2 Implemented Grip Gestures

Implementation of grip gestures was based on the results of the preliminary study.

The set of implemented grip gestures was defined by two factors: the results of the preliminary study (Section 4 "Preliminary Study: a Survey on Grasping Objects of a Cylinder Shape") and the possibilities of tracking with Vicon system. Since users expressed preference in using different areas of an object compared to a number of fingers in a grip, we implemented recognition of the following grip gestures: top, middle and bottom grips of the water bottle, and a tap on top of the bottle (Figure 5.2).

In addition, we used another everyday object, a chocolate bar, to show that touches on different parts of an object or a surface can also be mapped to controllers (Figure 5.3).

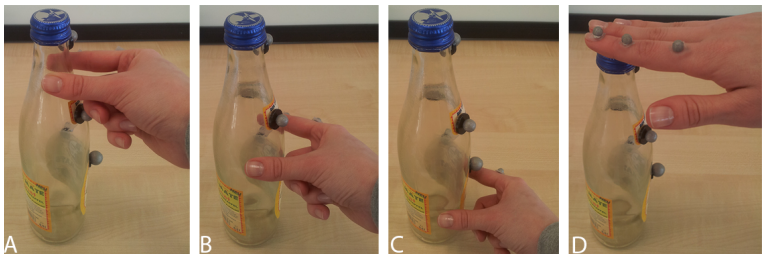


Figure 5.2: Three types of grip gestures were defined on the bottle object: A – top grip, B – middle grip, C – bottom grip, and a tap on top of the bottle (D) was also available for mappings.

Figure 5.3: Three touch-sensitive areas were defined on the chocolate bar object and were available for mappings.

5.3 User Action Sequences

Figure 5.4 shows an overview of the user action sequences for PBD and GUI-based applications. The interaction starts with the user input from the iPhone: the user selects *New connection* option from the menu, then he selects the device and the controller that he wants to map to the object (Figure 5.5).

The sequence of actions performed by users was the same in the PBD and GUI prototypes.

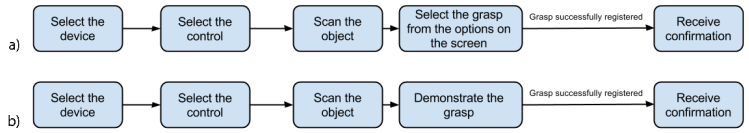


Figure 5.4: User action sequences: a) GUI-based approach, b) PBD approach. Both prototypes included the same sequence of actions for creating a mapping. The only difference was the way a grip gesture is selected or demonstrated.

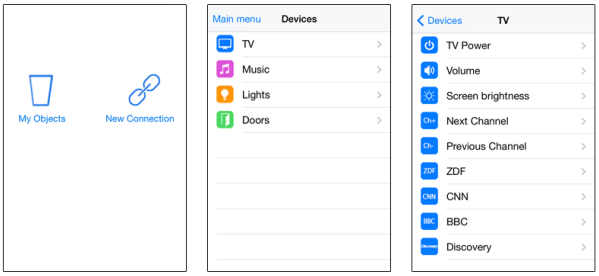


Figure 5.5: Interface for initiating the interaction, selecting the device and the controller was the same for the PBD and the GUI-based applications.

The current state of the object's mappings is presented to the user in both PBD and GUI applications after the object has been scanned.

Once the object has been identified by the system by scanning a QR code assigned to it, the current state of the object's mappings is presented to the user, e.g., that TV Volume controller is assigned to the bottom grip gesture (Figures 5.6 and 5.7).

During PBD users interact with objects directly.

Next, in case of PBD application the user can demonstrate the grip gesture on the physical object. Once the grip has been recognized, an interactive feedback in a form of a highlighted area is shown on the iPhone (Figure 5.6). If the user moves his hand to another area on the object (performs a different grasp) the system highlights the newly recognized area respectively and the user can confirm it by the tap or move his hand again and perform another grip. Once the user is satisfied with the performed grip gesture, he taps on the screen and a confirmation view appears to inform the user which mappings have been established.

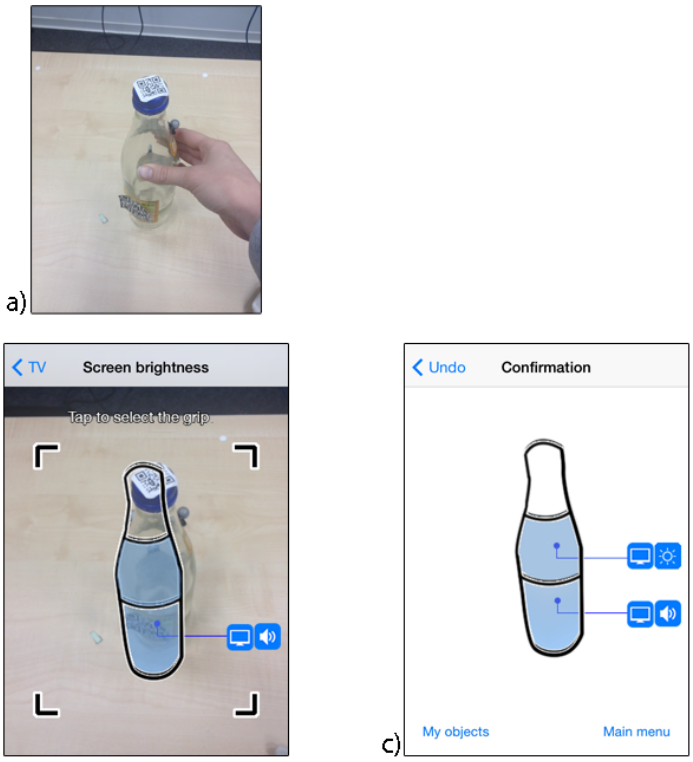


Figure 5.6: PBD application: once a user has identified an object to the system by scanning a QR code attached to it, the system presents which controllers are currently assigned to the object and to which areas (in this example, a TV volume controller is assigned to the bottom area or bottom grasp of the object). Then a user can demonstrate the grip gesture that he would like to map to the controller selected in the previous step (in this Figure middle grasp is being assigned to the TV Screen brightness control). The area corresponding to the performed grip gesture is highlighted on the screen (picture b). The interaction ends with a confirmation screen (picture c) after a user selected the grip gesture and taped on the screen.

In the GUI-based prototype, after the user scans the object, the system, similarly to the PBD approach, shows the current state of the object's mappings. Also the list of possible options for mappings with respect to the chosen object is presented on the iPhone screen (Figure 5.7). The user can then select the desired option and receive the confirmation.

In the GUI approach users select mappings from the virtual representations of grip gestures

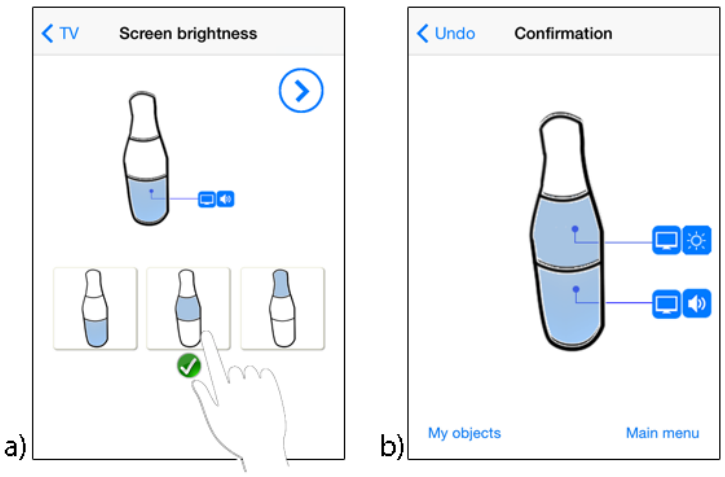


Figure 5.7: GUI-based application: once a user has identified the object to the system, the application presents the mappings that are currently assigned to the object (similarly to the PBD approach), then a user can tap on one of the icons to select a grip gesture that he would like to map to the selected controller (picture b). The interaction ends with a confirmation screen as in PBD application (picture c).

5.4 System Architecture

The developed software prototypes consisted of the following parts:

- Server application

The Server application coordinates data between the mobile application, Vicon system and TV and Lights control applications.

Two Server applications were developed for the PBD and the GUI-based prototypes respectively. The Server processes user input from the mobile application and it receives and interprets data stream from the Vicon tracking system. Once any mappings are created, the Server stores information about assigned controllers (object and the grip gesture) and later uses this data for activating the TV and Lights control applications (Figure 5.8).

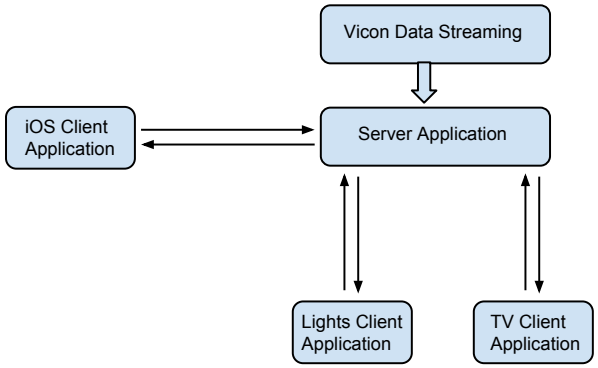


Figure 5.8: Overview of the system architecture: the Server processes user input from the mobile application and it receives and interprets data stream from the Vicon tracking system. When the mappings are created, the Server application sends translated values to the TV and Lights applications.

- iPhone application

  Users interact with the system through the iOS mobile application, we developed two such applications to support the PBD and the GUI end-user programming strategies. Users can use the smartphone for establishing the mappings, correcting and deleting it. Moreover, the application provides a reference of the controllers and grip gestures assigned to the objects.

  Users interacted with the system through the smartphone application.

- TV control application

  For one of the test scenarios we developed a Cocoa-based TV application. When users assigned TV-related controllers (power, volume, screen brightness, navigation between channels and programming a favorite channel) to an object, they could try out the interaction and control the TV with an everyday object. When users applied the respective grip gestures, the Server would interpret the data stream from the Vicon cameras, translate the values into the valid range and send it to the TV application for activation.

  For the user study we developed a sample TV application.

- Lights control application

  For another scenario we used Phillips Hue Smart Lights. Based on the provided API, we developed an

We integrated our system with Phillips Hue smart lights to present another scenario of using everyday objects as controllers.

application that allowed users to assign a light switch to an object, change the brightness and color of the lights and map three favorite color themes. For activating these controllers, the Lights control application needs to receive the values from the Server application (similarly to the TV scenario).

5.5 Data Communication

We used a networking framework that enables asynchronous calls for efficient data communication.

Efficient data communication between the parts of the implemented system was essential to provide interactive feedback and responsive experience. We used the ThoMoNetworking framework[2] developed at the Media Computing chair for network communication between the server and clients applications. The framework is based on the Bonjour architecture and provides automatic devices discovery, asynchronous calls and handling of the dropped connections. Data from the Vicon tracking system was streamed and handled by the Cocoa Server application. Below we describe the data communication in PBD and GUI-based prototypes in more details.

The interaction starts with the user input from the iPhone application, then the data is sent to the Server, that responds accordingly.

An abstract sequence diagram of data communication in GUI-based and PBD applications is shown in Figure 5.9. First, the user provides the system with input through the iPhone application: selected controller and object are sent to the Server. Based on this information the Server can send the iPhone application the information about the selected object, i.e., if any controllers and grasps have already been assigned to this object. This information is then visualized in the iPhone application and presented to the user.

Next, in case of the GUI-based application the user can select one of the options from the list on the screen to choose the desired grip gesture (area on the object). His choice is then sent to the Server and the newly created mapping is stored there.

[2] http://hci.rwth-aachen.de/thomonet

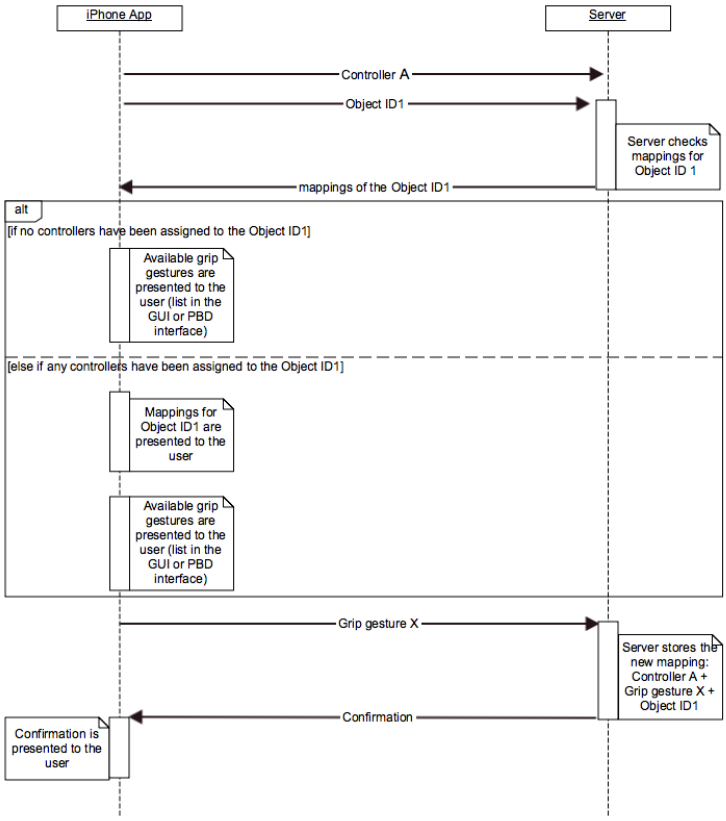


Figure 5.9: Data communication in PBD and GUI-based applications: after the user selects the object, the Server checks if any mappings are currently assigned to it. If none of the controllers are currently assigned to the object, the user is provided with the interface of selecting or demonstrating a grip gesture, and if existing mappings are found, they are presented to the user in the mobile application and then the user proceeds to assigning a grip gesture.

Data communication in PBD approach includes interaction with the Vicon system.

In case of the PBD application, data communication is more complex and requires connection to the Vicon tracking system (Figure 5.10). When the controller and an object are selected, the system enters *demonstration mode*, in which the Server is listening to the data stream from Vicon system, interprets it and later maps it with the selected controller. This stream consists of the marker's X,Y and Z coordinates and some additional derived data about the segments.

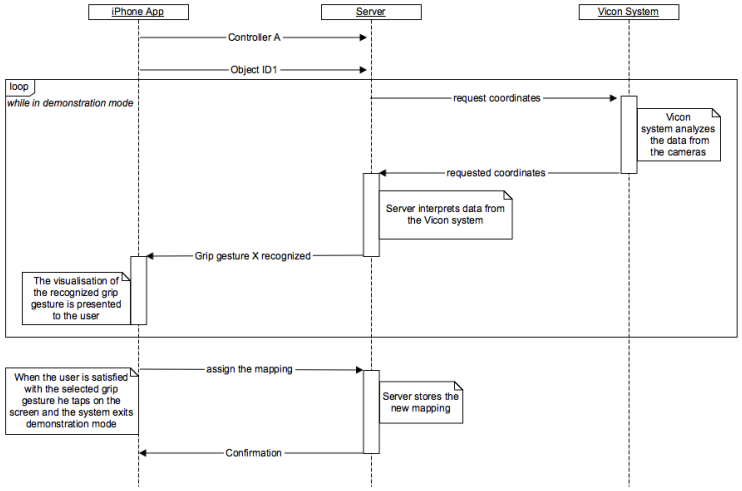


Figure 5.10: Data communication in the *demonstration mode*: the Server is listening to the data stream from the Vicon system, interprets it and later maps it with the selected controller.

PBD prototype had two modes: *the demonstration mode*, during which the Server was interpreting the data stream from Vicon system as grip gestures that could be mapped to the selected controller, and *the use mode*.

During the study participants were asked to wear a glove that was augmented with reflective markers to enable tracking of the grip gestures. While the user was demonstrating a grip, the Server was interpreting the data based on the position of user's hand relative to the object. Once the Server recognized a certain grip gesture, e.g., a bottom grasp of the bottle, it sent a message to the iPhone application that would inform the user about the current grasp by highlighting the respective area on the screen. This data communication is repeating while the user is interacting with the object, this way, interactive feedback is provided. The mapping is stored on the Server once the user is satisfied with his choice and has tapped on the screen. Then the iPhone application sends the Server a message to stop interpreting the data stream for the chosen controller and save the mapping.

In the *use mode* the Server was interpreting the data stream from Vicon system and activating a TV or Lights application with respect to the stored mappings.

When the mappings are established, the system is in the *use mode*, i.e. when the user performs the motion with the object applying the assigned grip gesture, the value of the respective controller changes accordingly. Figure 5.11 shows the data communication between the Vicon tracking system, Server, TV and Lights applications in the *use mode*.

Server looks up which controllers and grip gestures are currently assigned and listens to the data from the Vicon system. If a match for the performed grip gesture is found, Server translates the coordinates data into the values for the controllers and sends it to the TV or Lights application respectively.

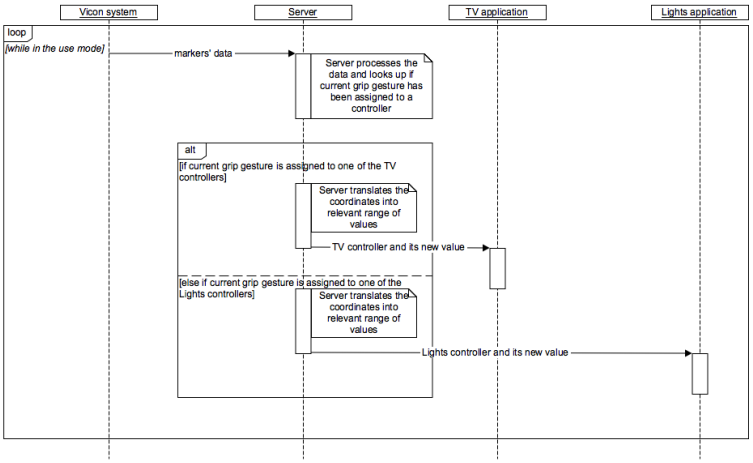


Figure 5.11: Data communication in the *use mode*: the Server listens to the data from the Vicon system. If a match for the performed grip gesture is found, Server translates the coordinates data into the values for the controllers and sends it to the TV or Lights application respectively.

Chapter 6

Interactive User Study: Programming Everyday Objects as Controllers using Grip Gestures

6.1 Study Design

The goal of the user study was to answer the following research question:

- Is *Programming by Demonstration* preferred over the *GUI-based approach* for programming everyday objects using grip gestures?

The goal of the study was to find out if PBD is preferred over the GUI-based approach for programming everyday objects using grip gestures.

The null hypothesis is that there is no statistically significant preference in programming everyday objects using grip gestures between the programming by demonstration and programming by choosing the options from the Graphical User Interface.

We conducted a within-group user study, where independent variables (IV) were two strategies for programming everyday objects: *by demonstration* and programming by

choosing the options from the *Graphical User Interface*. Dependent variables (DV) were the following usability characteristics partly based on the System Usability Scale (SUS): *understandability, ease of use, learnability, support for expressing user's intentions, ease of assigning binary and continuous controls, making corrections and visualization of previously assigned grasps*. Other DV are characteristics more specific to the presented prototypes: *visualization of the current and previously assigned grip gestures* and *feasibility of mapping gestures with more degrees of freedom* (full description can be found in Section 6.4 "Evaluation Methods").

6.2 Setup

The setup for the study included Vicon system for tracking grip gestures.

The setup for the user study consisted of an array of eight Vicon Bonita cameras positioned around the tracked area. To enable tracking of an object and recognition of grip gestures, objects and user's hand were augmented with reflective markers. Users programmed everyday objects with the iPhone applications, that we developed for the GUI-based and PBD approaches respectively. Sample tasks included controlling the TV application and adjusting lights with Phillips Hue smart lights.

6.3 Procedure

Users were asked preliminary information including information about their handedness.

In the beginning of the study users were asked preliminary information about their age, occupation and familiarity with using iPhone applications. We also asked users about their handedness, as it could have affected the interaction. Then users were asked to perform a set of tasks using both prototypes. Further in this section we describe the scenarios, tasks and evaluation methods that we used in the study.

Figure 6.1: The setup included eight Vicon Bonita cameras, objects with reflective markers, iPhone applications for the PBD and GUI approaches, TV application and Phillips Hue smart lights. User's hand was also augmented with three reflective markers.

6.3.1 Scenarios

Users were presented two scenarios including TV and Lights control.

During the study users were presented two scenarios with tasks that could take place in a living room – a TV control scenario and a lights scenario. In each scenario the user had two objects – a bottle and a chocolate bar – and he used different grasps in order to assign multiple controls to it. Scenarios also reflect the possible motivation for using everyday object as a controller: when the dedicated controller is unavailable (Scenario 1) and for better convenience compared to the dedicated controller (Scenario 2).

Scenario 1 - TV Control The user's TV remote control is unavailable (e.g., it has been misplaced or its batteries are empty), therefore he reprograms objects around him to substitute a desired functionality:

- Volume control
- Screen brightness control
- TV power
- Three favorite channels

- Next channel
- Previous channel

Scenario 2 - Lights control The user got new batteries for his remote control and he doesn't need to use EDO for controlling his TV anymore. But instead he decides to use EDO to control lights in the living room since the light switch is located on the other side of the room. For convenience the user assigns the following light controls to the objects he has on the table:

- Lights On/Off switch
- Dimming the lights
- Changing the color of the lights
- Programming favorite light theme (up to 3)

6.3.2 Tasks and instructions

Users were given instructions for assigning specified mappings.

Since the focus of the study was not to investigate which grip gestures would the user choose for different mappings, but how he communicates the desired behavior to the system, the users were given precise instructions on which grasps to use for mapping the controllers. The instructions are listed in Tables 6.1 and 6.2.

The order of tasks was defined by a Latin square design.

For minimizing order effect we applied a *Latin Square* design to the sequence of prototypes presented to users (users started either with the PBD or GUI-based prototype), to the sequence of scenarios presented to the user (the TV or Lights control scenario) and to the sequence of tasks within each scenario (the tasks listed in Tables 6.1 and 6.2. are independent from each other and can be used in a Latin square design).

TV Scenario	
Object: Bottle	1. Assign *Power Control* to the *tap on top* of the bottle 2. Assign *Volume Control* to the *bottom grasp* of the bottle 3. Assign *Screen Brightness Control* to the *middle grasp* of the bottle, then change *Screen Brightness control* to the *top grasp*
Object: Chocolate Bar	1. Assign *Switching to Next and Previous Channels* to the *first* and *second segments* of the chocolate bar 2. Delete connections for *Navigation between Channels* and program *3 favorite channels (1, 2, 3 segments of the chocolate bar: 1 - CNN, 2 - ZDF, 3 - BBC)* 3. Delete *CNN Channel* as favorite and substitute it with *Discovery Channel* After finishing all tasks from TV Scenario *delete all mappings*

Table 6.1: Tasks and instructions for the user - TV control

Lights Scenario	
Object: Bottle	1. Assign *Switching the Lights On and Off* to the *tap on top* of the bottle 2. Assign *Brightness Control* to the *bottom grasp* of the bottle 3. Assign *Color Control* to the *middle grasp* of the bottle
Object: chocolate bar	1. Assign *3 Favorite Lights Themes* to the segments of the chocolate bar: *first segment - "Lime theme", second segment - "Retro theme", third segment - "Sunset theme"* After finishing all tasks from the Lights Scenario *delete all mappings*

Table 6.2: Tasks and instructions for the user - Lights control

6.4 Evaluation Methods

The following qualitative evaluation methods were used:

- Observation combined with *Think-aloud*

  Participants were asked to comment on their actions during working on the tasks and investigator was observing and taking notes for future investigation.

- Questionnaires

  Users were asked to fill in two types of questionnaires – one was designed to evaluate general usability characteristics of the prototypes and the second type was focusing on comparison of the presented approaches. Further description of the questionnaires is presented in Section 6.4.1 "Questionnaires"

- Interview

  Another technique used for evaluation was the interview that was conducted at the end of the study. The goal of the interview was to have a discussion with the user about his experience and let him express his feedback in a form of open-ended questions.

6.4.1 Questionnaires

The first type, Questionnaire A, was based on the System Usability Scale (SUS) and additional questions specific to the prototypes. We asked users to fill it in after completing the tasks with each – the GUI and PBD – prototype. Users were asked to answer the questions in a form of Likert Scale. The questions were grouped to evaluate the following usability characteristics:

- Understandability
  - The features of the system are comprehensive for me.

  - I found the sequence of actions reasonable for achieving my goal.

- Ease of use

  - I found the system unnecessarily complex.
  - I found the system easy to use.

Questionnaire A was designed to evaluate the overall usability characteristics.

- Learnability

  - I would imagine that most people would learn to use this system very quickly.
  - I needed to learn a lot of things before I could get going with the system.

- Expressing user's intentions

  - I felt confident using this system.
  - The system allowed me to express my intentions.

- Ease of assigning binary controls

  - I found it easy to assign binary controls (push button) to an object using this system.

- Ease of assigning continuous controls

  - I found it easy to assign continuous controls (rotary knob) to an object using this system.

- Making corrections

  - I found it easy to make corrections in the mappings between objects and controllers.

- Visualization of previously assigned grasps

  - The visualization of previously assigned grasps (intermediate step during programming an object) was clear and helpful in achieving my goal.

Questionnaire B included questions that emphasized the differences between the PBD and GUI approaches.

Questionnaire B consisted of the questions specific to the presented approaches and their implementation. Participants of the study were asked to fill in Questionnaire B after they tried both prototypes. Full texts of the questionnaires are included in Appendix B "Appendix for the Interactive User Study".

6.5 Participants

We received 14 participants, aged 20-29 (M = 26, SD = 2.51, six females) for this user study. Two of the participants were Mechanical Engineering students, others were Computer Science students or recent graduates. All the participants were right-handed, all except two had prior experience with an iPhone.

6.6 Evaluation

We were able to get interesting insights and feedback from users expressing individual preferences.

In this section we present the results of Observation, Questionnaires and Interviews conducted during the final user study. The results did not show a statistically significant preference towards the GUI or PBD approach for programming everyday objects, however, users evaluated the presented prototypes rather positively and they expressed individual preferences.

6.6.1 Observation Results

Users switched hands during the PBD interaction

During demonstration on the object users had a possibility to hold the phone in a free hand or put it down on the table.

For the PBD condition, users had a possibility to hold the phone in their hands all the time or to put in down on the table during the demonstration on the object. Users (all right-handed) started by holding the phone in the right hand and when it was time to perform a demonstration on the object they needed to free the right hand and they put the phone either in the left hand or on the table. In the beginning most of the users (11) held the phone in their left hand during the demonstration, but later they found it more convenient to put the phone down on the table. In both cases, the user needed to switch hands during the interaction.

Participants performed the tasks faster with the GUI-based application

Users felt that the GUI approach allowed them to complete the tasks faster.

Users performed the tasks faster with the GUI-based application compared to the PBD application. This statement comes from subjective estimation of the examiner, but it was also confirmed by the questionnaire (Section 6.6.2 "Questionnaire B"). We noticed that participants did not check with the object when choosing a grip gesture from the list in the application – this saved them time compared to the PBD approach. Once the participants got familiar with the sequence of actions in the application, they looked more confident while programming the object with the GUI application. This can be explained by a familiar type of interaction with a smartphone.

Frequent need to scan the object made the sequence interruptive

Once an object is scanned it should be remembered by the system.

In both GUI and PBD prototypes users were choosing the object for interaction by scanning the QR code attached to it. Observation and comments from the participants have shown that 10 out of 14 users found it inconvenient to scan the QR code of the object every time when they wanted to map a new control. Users were expecting the system to remember the object once they scanned it. This observation was also confirmed by the questionnaire: when users were asked their opinion about the sequence of actions six participants expressed that the frequent need to scan the QR code of the object was making the interaction interruptive.

Some users found the Undo function inconvenient

While the users were interacting with the prototypes we noticed that after getting to the confirmation screen and then being asked to program another control, several users were reaching to the top left of the screen as if they wanted to go back. One user mentioned that "Using back navigation for Undo felt awkward". One reason for that could be that it was not clear for the user where to go from the Confirmation screen. A possible solution would be to provide clearer next step, such as e.g. clearly visible "Done" button. Another solution proposed by one of the users was to

substitute the Undo with simple back navigation (without canceling the last action) and from that view give the user a possibility to add more controls to the same object. In order to provide the Undo functionality a separate button or a quick shake of the iPhone can be used as suggested by the *Apple iOS Human Interface Guidelines*.

Names of the menu items were not clear

Users made some general comments about the mobile applications.

Five users commented that the naming in the application was not entirely clear for them. "New connection" was not associated with the start of the object-programming sequence and "Main menu" was found confusing too.

This concludes our findings from the observation and now we will examine the results of the questionnaires to get further insights about users' feedback.

6.6.2 Questionnaire Results

As described in section 6.4.1 "Questionnaires" two types of questionnaire were presented to the user: the first one - Questionnaire A – aimed to evaluate the PBD and GUI-based prototypes on usability characteristics. The second type – Questionnaire B – was focusing on the differences between two designed applications. Both questionnaires are included in the Appendix B "Appendix for the Interactive User Study".

Questionnaire A

The results were analyzed with Wilcoxon Signed Rank statistical test.

Questionnaire A evaluated such parameters as *Understandability, Learnability, Ease of Use, Expressing User's Intentions, Ease of Assigning Binary and Continuous Controls, Making Corrections and Visualization of Previously Assigned Grasps*. We analyzed the results by performing *Wilcoxon Signed Rank* statistical test.

Understandability

All 14 users agreed or strongly agreed that the features of each application were comprehensive to them (M_{GUI} = 4.57, SD_{GUI} = 0.51, M_{PBD} = 4.50, SD_{PBD} = 0.52, n.s.): it was clear how to choose the controller, how to scan the object and assign the mappings (Figure 6.2).

Users found the features of both prototype systems comprehensive.

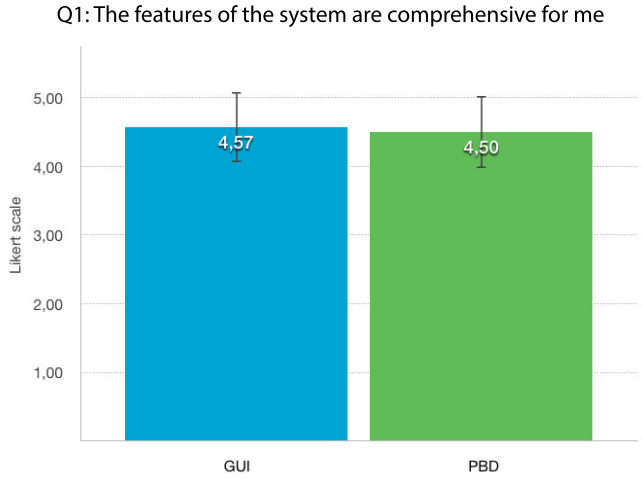


Figure 6.2: Users found the features of the GUI-based and PBD applications comprehensive.

The sequence of actions was the same in both applications and was rated equally for both approaches as reasonable (M_{GUI} = 4.14, SD_{GUI} = 0.54, M_{PBD} = 4.36, SD_{PBD} = 0.50, n.s.). However, based on the users' comments the sequence of actions could be optimized by eliminating the need to scan the object every time a new controller is chosen and by having a possibility to start from choosing the object rather than from choosing the device. The first point was expressed by 10 users and one possible solution could be to store the object in the system once it has been scanned and later let the user pick it from the app. Four users mentioned that in some cases they would like to start from choosing the object and map several controllers to it. Determining the optimal sequence was not the goal of this study, however the described findings could be used for further investigation.

Users found the sequence of actions reasonable and expressed possible improvements.

Ease of use

Users found the GUI and PBD applications easy to use.

Figure 6.3 demonstrates that both applications were rated as easy to use (M_{GUI} = 4.36, SD_{GUI} = 0.63, M_{PBD} = 4.29, SD_{PBD} = 0.61, n.s.) and users disagreed that applications were unnecessarily complex (M_{GUI} = 1.57, SD_{GUI} = 0.76, M_{PBD} = 1.86, SD_{PBD} = 0.54, n.s.).

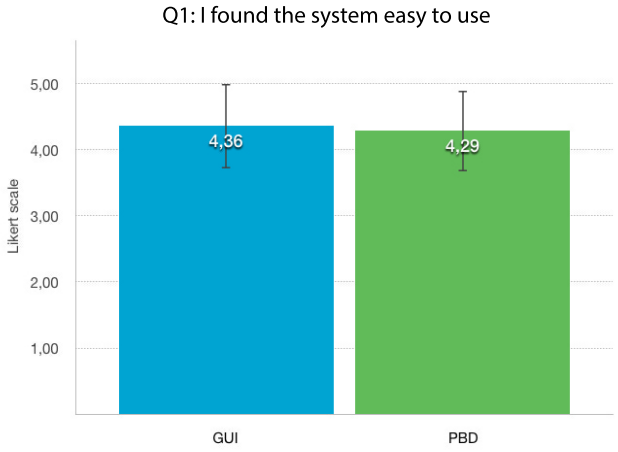


Figure 6.3: Users rated the GUI-based and PBD applications as easy to use.

Learnability

GUI and PBD approaches were found easy to learn.

All participants agreed or strongly agreed that most people would learn to use each application — GUI and PBD — very quickly (M_{GUI} = 4.29, SD_{GUI} = 0.61,M_{PBD} = 4.29, SD_{PBD} = 0.73, n.s.). Neither of the participants agreed that he needed to learn a lot of things before he could work with the GUI or PBD applications (M_{GUI} = 1.50, SD_{GUI} = 0.52, M_{PBD} = 1.64, SD_{PBD} = 0.63, n.s.).

Expressing user's intentions

Users expressed that they felt rather confident using both systems.

There was no significant difference in how confident the user felt using GUI-based or PBD application (M_{GUI} = 4.50, SD_{GUI} = 0.65, M_{PBD} = 4.29, SD_{PBD} = 0.61, n.s.). Figure 6.4 shows that participants agreed that both applications allowed them to express their intentions (M_{GUI} = 4.14, SD_{GUI} = 0.87, M_{PBD} = 4.29, SD_{PBD} = 0.47, n.s.). In future work,

PBD approach could support machine learning algorithms and let users teach the system, in this case users would have more freedom in expressing their intentions.

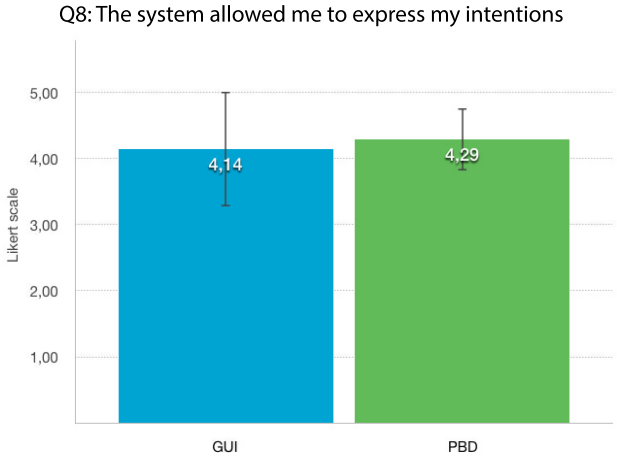


Figure 6.4: Both GUI-based and PBD applications allowed users to express their intentions.

Ease of assigning binary and continuous controls

Users did not experience problems with assigning binary or continuous types of controls.

All users found it easy to assign binary controls with the GUI-based and PBD applications (M_{GUI} = 4.43, SD_{GUI} = 0.51), M_{PBD} = 4.43, SD_{PBD} = 0.51, n.s.). Users also did not report any problems about assigning continuous controls (M_{PBD} = 4.43, SD_{PBD} = 0.65, M_{GUI} = 4.43, SD_{GUI} = 0.51, n.s.).

It is important to emphasize (and it was communicated to the participants) that the resolution of the mapping, e.g., which position of the object should correspond to the minimum and maximum was not evaluated in this question. In the developed prototypes we implemented natural mappings, i.e., for increasing a property (such as volume or brightness of the lights) the user rotated the object clockwise and for decreasing it — counterclockwise (analogously to a physical knob), for mapping the color we used the spectrum from violet to red. Although users did not express any confusion regarding the implemented mappings, more investigation is necessary in order to make any conclusions.

Making corrections

Users were able to make corrections in mappings and undo any performed action.

In this question we evaluated how the user could correct his choice of mappings: e.g., if it was possible to undo the connections between an object and a controller. The results show that all the users found it easy to correct their actions in both PBD and GUI-based applications (M_{GUI} = 4.50, SD_{GUI} = 0.52, M_{PBD} = 4.43, SD_{PBD} = 0.52, n.s.). However, as it was already described in the results of the observation, some users were confused with using back navigation as Undo function. Discussion of alternative solutions can be found in section 6.6.1 "Observation Results".

Visualization of previously assigned grasps

Visualization of previously assigned to the object controllers helped users to make decisions about future mappings.

The results (Figure 6.5) show that all the participants of the study found the visualization of the previously assigned mappings helpful in both GUI-based and PBD approaches (M_{GUI} = 4.57, SD_{GUI} = 0.51, M_{PBD} = 4.50, SD_{PBD} = 0.52, n.s.). Users were able to interpret it without any problems.

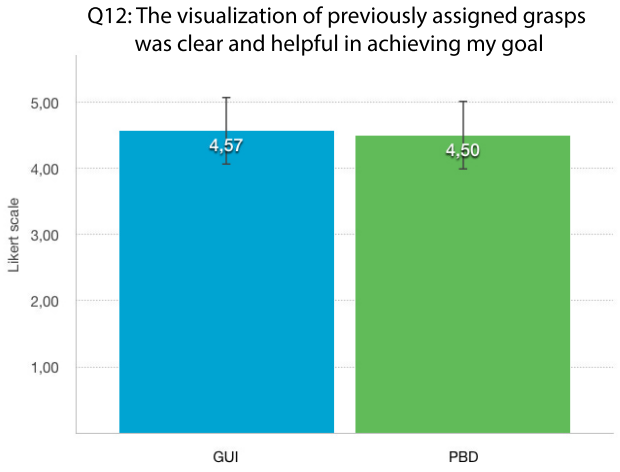


Figure 6.5: Visualization of previously assigned to an object grasps and controllers was clear and helpful to users in both GUI-based and PBD approaches.

Questionnaire B

Questionnaire B was designed to compare the PBD and GUI-based approaches and better understand users' preferences. We saw that, although, there was no clear preference towards one programming strategy, we were able to identify individual advantages and drawbacks of each approach.

We present the results of this questionnaire in a form of *diverging stacked bar charts* introduced by Robbins and Heiberger [2011]. This representation shows the percentage of respondents who agreed with a statement on the right side of the common baseline, and the percentage of people who disagreed with a statement on the left side. The percentage of people who neither agreed nor disagreed with a statement is positioned in a way that it is split in half by the zero baseline. The plots were build using HH Package [1] in R software.

For better visualization the results of this section are presented in a form of diverging stacked bar charts.

Visualization of the current grasp and previously assigned mappings

There was no clear preference in the visualizations of the chosen grasp and previously assigned mappings between the PBD and GUI-based applications: three users chose the GUI prototype arguing that "it was clearer which grasps are available", three users chose the PBD approach saying that they prefer to "show a grasp to the system". Moreover, two users who chose PBD in this question, mentioned that a projection or "additional clues" on the object could be helpful. Two other participants said that for more complex grasps they would choose PBD visualization, as the GUI-based one may not be able to express the angle or the exact position of the grasp, but for simple mappings as in the developed prototype the GUI application was equally preferred. Another six users found both visualizations equal: for instance, one of the users said that both approaches have their advantages – the GUI-based application is "easier to handle", but the PBD approach "lets you know right away how the grasp feels".

The description and illustration of visualizations can be found in Section 5.3 "User Action Sequences".

[1] http://cran.r-project.org/package=HH

Identification of available grasps

Such criteria as easy identification of available grasps is valid only for prototypes where the set of grip gestures or mappings is limited by the system.

As shown in Figure 6.6, nine users agreed or strongly agreed that it was easier to identify available grasps with the GUI-based application. One of the users mentioned that "with the GUI application I could see right away which grasps can be used, whereas in the PBD I needed to explore it myself". Another comment supporting the idea of easier identification of available grasps in the GUI-based prototype compared to the PBD is, as one of the participants said, that "it was not clear how accurate I should be when grasping the bottle in the PBD application". This gives us an idea that the user had to try out different hand positions before he could say which grasps are recognized — this problem could be eliminated if the system automatically recognized any grasp performed by the user (then the user would not have to adjust to the system, but the system would adjust to the user).

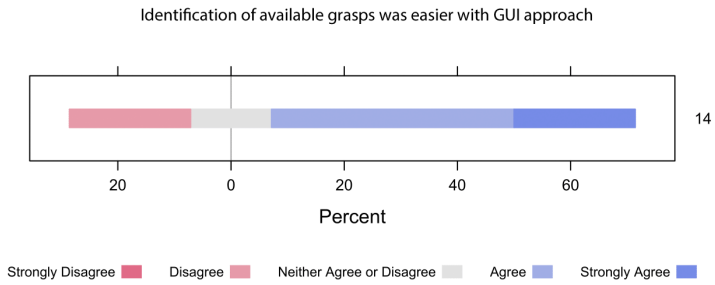


Figure 6.6: Results show that users found it easier to identify available grasps with the GUI-based prototype.

In PBD approach, if the set of grip gestures available for mappings is limited, the possibilities should be clear to the user.This could be visualized on the smartphone or by projecting visual clues directly on the object.

Three users stated that with the current implementation even in the PBD application it was still easy to recognize which grasps can be used, however they pointed out that it might change if more options are available. Overview of the grasps or areas on the object available for programming helps the user understand how many controllers he could assign and how, this information was not constantly present in the PBD application (the user had to move his hand around the object in order to get the feedback from the system), but it could be provided, e.g., by highlighting areas available for mappings.

Applying the grasp after programming the object with PBD

It was easier for users to apply a grip gesture for controlling a device after they "tried it out" during the demonstration of the grip gesture on an object.

Participants (nine) found that applying the grasp to the object for controlling a device after programming it with PBD felt more natural (Figure 6.7). A comment supporting this idea was that during the demonstration in PBD approach users could "try out" the grasp, that gave them clearer understanding of how to perform the grasp when controlling a device.

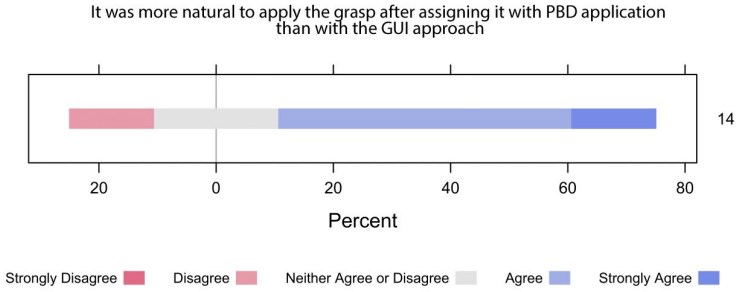


Figure 6.7: Nine users agreed that applying the grasp after programming the object with the PBD felt more natural compared to the GUI-based application.

The effort spent during PBD

Most of the users did not find the PBD approach harder, and those who did, mentioned that the cause was, mostly, the need to interact with the smartphone before and after performing the demonstration on the object.

When participants were asked if they found it harder to demonstrate the grasp in the PBD approach compared to picking it from the options in the GUI application, only six people agreed or strongly agreed with this statement (Figure 6.8). From the comments we can see that one of the reasons for PBD to be harder was that the user often needed both hands during the interaction, e.g., he first selected a controller from the smartphone application, then scanned the QR code of the object, and then the user could either put the phone on the table and perform the grasp or put the phone in the other hand and demonstrate the grasp with the other. As shown in the section 6.6.1 "Observation Results" most of the users started with holding the phone in their hands during the demonstration, but later they found it more convenient to put the phone on the table. An additional consequence of this interaction is the need for the

user to switch his attention between the smartphone and the object. We will discuss this later in the section (Subsection "Switching attention during PBD").

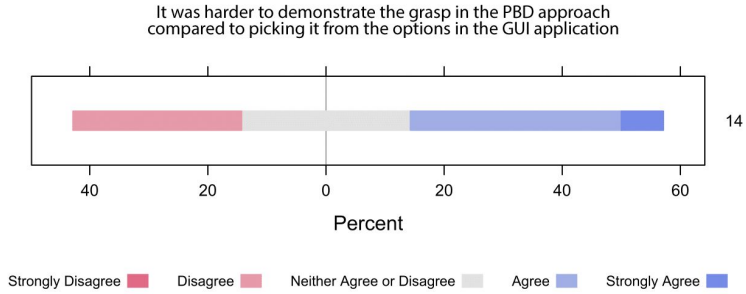


Figure 6.8: When users were asked if they found PBD harder compared to the GUI-based approach, only six users agreed with the statement. The remaining eight users either disagreed that PBD interaction was harder (four participants), or did not agree or disagree with the statement.

The novelty of the PBD approach compared to a more familiar GUI-based interaction also affected users' preferences.

Another reason why some users found the interaction with PBD harder was its novelty compared to the familiar typical interaction with the smartphone. One of the users expressed his opinion as follows: "I am pretty used to the GUI. I had to use smartphone in both prototypes anyway. GUI itself worked pretty good". This user believed there was an extra effort he had to exert in the PBD approach, while he could achieve his goal with a more familiar GUI-based interaction.

However, the remaining eight users either disagreed that PBD interaction was harder (four participants), or neither agreed or disagreed with the statement.

The time spent during PBD

Users felt that they completed the tasks faster with the GUI-based approach.

Eight users found the PBD approach more time-consuming compared to the GUI-based application (Figure 6.9). One of the reasons (expressed by five people) was the need to switch the attention between the smartphone and demonstration on the object. Users said that it took some time "to check on the phone that my grasp was correct".

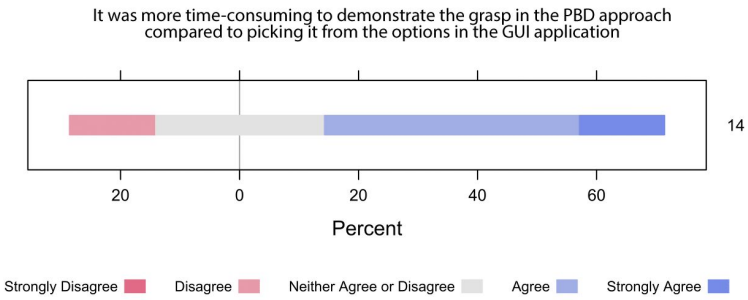


Figure 6.9: Eight users found the PBD approach more time-consuming compared to the GUI-based approach. One of the reasons was the need to switch the attention between the smartphone and demonstration on the object.

Participants were not asked to assign the mappings as fast as possible and in fact some users "played" with the PBD application for a while before assigning the grasp. For two users this was the reason why they found PBD more time-consuming.

Switching attention during PBD

As our implementation of the PBD approach included interaction with a smartphone along with demonstration, eight users found switching attention interruptive.

Switching attention between the smartphone and demonstration on the real object was found interruptive by eight users (Figure 6.10). They expressed that in the beginning, they were surprised to interact with the real object, one of the users said that he was "so used to click "next" in the app (from his everyday interaction with the phone) that he did not expect to switch to an object". Other users added that the interruption was caused by checking the feedback on the screen and making sure it corresponded to the demonstration. Switching attention was one of the main inconveniences that participants have experienced with PBD application, it required more time and effort from the user compared to the GUI-based approach. We foresaw this problem and therefore in the interview we raised this question again to better understand how the issue could be resolved (see section 6.6.3 "Interview results".

On the other hand, one of the users who found switching attention interruptive mentioned that despite this fact, the PBD application could be a better fit for a situation when

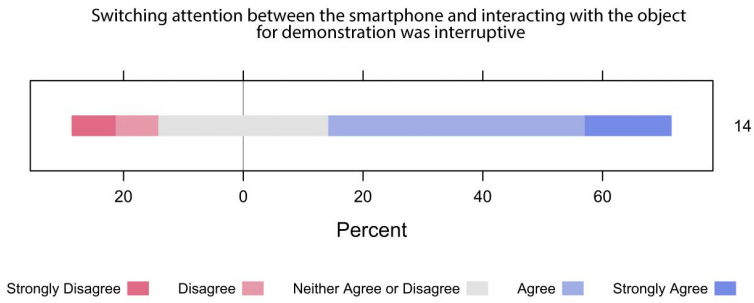


Figure 6.10: Switching attention between the smartphone and demonstration on the real object was found interruptive by eight users.

Despite the interruptiveness of attention switching, PBD was considered a promising approach especially for assigning more complex gestures.

the user can map more complex grasps or even movements of objects. This points to a trade-off between the expressiveness (which can be achieved with the PBD approach) and non-interruptive and familiar interaction with the smartphone.

Users who did not find it interruptive explained that switching attention is something that people experience in their everyday life and therefore learn how to handle it, for example, while operating a TV with a remote control and looking at the TV screen.

Feasibility of assigning grasps with more degrees of freedom

We wanted to get an insight of user's preferences for programming objects with more complex grip gestures.

The current prototype implementation allowed users to assign controllers to a defined set of areas on the objects (four for the water bottle and three for the chocolate box), however, we wanted to get an insight of user's preferences in a case when more options of how the grasp can be applied are available. For instance, an angle of a grasp, a size of the contact surface, a number of fingers (e.g., using two or five fingers in a grip), or a performed motion with an object. Therefore, in the last question of the questionnaire we asked user's opinion about the feasibility of using PBD and GUI-based applications in scenarios where more degrees of freedom can be applied to an object. The results (Figu-

re 6.11) show that users found it feasible to map grasps or motions with more degrees of freedom with both PBD approach (12 users) and the GUI-based application (10 users). The visualization a form of *divergent stacked bar charts* allows clear comparison of the approaches.

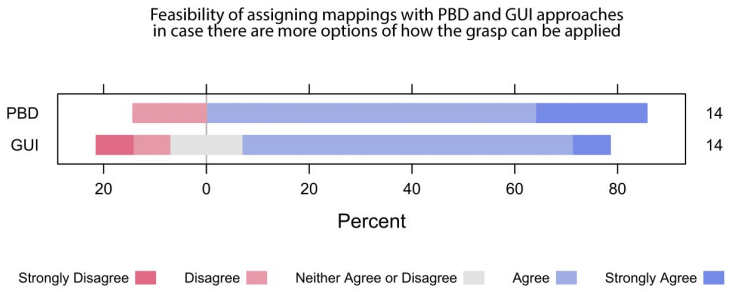


Figure 6.11: Users found it feasible to map grasps or motions with more degrees of freedom with both the PBD approach (12 users) and the GUI-based application (10 users).

The main argument in favor of the GUI-based approach was the visibility of available mapping possibilities and a familiar interaction with the smartphone. However, when users were asked to express their preference between the two approaches for this task, eight users chose PBD and only three preferred the GUI-based application (Figure 6.12).

Eight users said they would prefer the PBD approach over the GUI-based one in cases with more complex gestures.

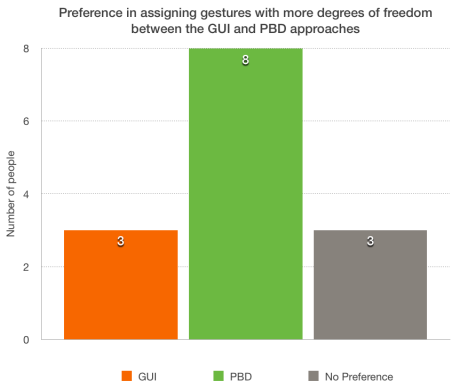


Figure 6.12: Results of users' preference between the GUI-based and PBD approaches for assigning gestures with more degrees of freedom.

One of the main advantages of PBD is its expressiveness – it could allow users to demonstrate and map any creative gesture performed with an object.

The motivation for the PBD approach was its expressiveness – a user could demonstrate any grasp or motion with the object, which could be hard to be represented in a GUI application. Users also mentioned that during the demonstration they got a chance to experience and "try out" the grasp before assigning it and it would be even more useful in the scenario where more degrees of freedom can be applied to a gesture with an object.

6.6.3 Interview results

After users filled in the last questionnaire they were invited to a short interview where they could share their impression of the prototypes. Participants were asked to express what they found good or bad, and which possible improvements they saw. We also went back to two questions from the questionnaire to ask the users to elaborate their choices. Those questions were regarding the attention switching between the smartphone and demonstration on the object during PBD and the feasibility of assigning more complex mappings with the introduced approaches.

Attention switching

Users suggested Voice control as an alternative to using a smartphone in PBD approach to avoid attention switching.

We asked the users who found it interruptive to switch attention between the smartphone and demonstration on the object if they saw any alternatives to the smartphone for the PBD approach. One of the suggestions was to use voice control for choosing a controller (e.g., by saying "TV Volume") and then demonstrate the grasp on the object. In this case special voice commands or gestures would be necessary for canceling or deleting a mapping. Moreover several users pointed out that a projection on the object would be helpful during the programming and after the mappings are established, since there would be no list of mappings stored on the smartphone. We also suggested such options as external monitor, menu projection on the table surface or on user's hand, but all the users found those alternatives equal or worse than the smartphone solution.

Feedback on the object

Users expressed difficulty to remember the mappings assigned to an object.

One of the common notes stated by the participants of the study was the difficulty to remember which controller was assigned to which area. Although users had a list of the created mappings in the smartphone application, they found it inconvenient to turn to a phone every time they needed to know which functionality has been assigned to an object. All the users agreed that the projected feedback on the object would be helpful.

Feasibility of assigning grasps with more degrees of freedom

Complex gestures and motions might be hard to be represented on a smartphone screen.

As described in section 6.6.2 "Questionnaire Results" users found it feasible to assign mappings with more degrees of freedom with both the PBD and the GUI approaches. The interview results supported that statement and further we discuss it in more details. The GUI approach was considered feasible due to a familiar type of interaction and clear visibility of available options for mappings. Five users noted that in case of more mapping possibilities it might be hard to represent all the options on a relatively small smartphone screen. Moreover, if we add motions with objects as another possibility for mappings — it could be hard to visualize such interactions in a clear way. However, users also pointed out that it depends on the scenario and that in some cases simple mappings to the object's areas are enough. For instance if the task is to map game controllers for moving the character forward, backward, left and right and the user decided to repurpose a water bottle for it, then during the programming he would not have to map each movement individually, but he would just indicate the area on the object (e.g. top grasp) and the natural mappings would be applied automatically.

Systems that use everyday objects as controllers in creative domains can benefit from PBD due to its expressiveness.

However, PBD gives users more freedom in interactions and could be a winning approach in creative domains. For example, if an application allows users to map motions with objects to music composition, or to create individual game controllers including such interactions as passing an object to each other or applying deformations to it (e.g., squeezing, stretching, blowing, etc.).

6.7 Findings

The conducted study showed how PBD and GUI-based approaches can be used for programming everyday objects using grip gestures. First, we evaluated implemented applications on *Understandability, Ease of use, Learnability, Expressing user's intentions, Ease of assigning binary and continuous controls, Making corrections and Visualization of previously assigned grasps*, then we conducted another survey study and an interview in order to better understand the benefits and limitations of each approach.

Both PBD and GUI prototypes received good feedback in terms of usability characteristics.

The results have shown that users could successfully assign different grip gestures to an object for controlling digital devices with both PBD and the GUI-based applications. Users found both approaches easy to use, understandable and easy to learn. Participants also stated that they were able to express their intentions and achieve their goals with both applications. Making corrections was also possible in both prototypes and users could effortlessly undo their actions.

Projecting assigned to an object mappings as well as indicating areas on an object available for programming could improve the experience from the interaction.

Visualization of previously assigned grasps was found clear and helpful – the user could interpret which controllers have already been mapped to the object. However, a possible improvement would be to use a **projection on the object for the feedback**. For instance, users noted that it was hard to remember which controllers have been assigned to the object and looking it up in the smartphone application was rather inconvenient. A projection on the object informing the user which areas of the object are in use and by which controllers could improve the user experience. Moreover, if the system is able to recognize only some areas or grip gestures, then the projection could be used to inform the user of available mapping options. This functionality could be useful for the PBD approach since users mentioned that it was easier to identify available grip gestures with the GUI approach, whereas in the PBD application they had to explore the options themselves by moving the hand along the object.

Limitations of the PBD approach

Including smartphone as an intermediate device provided an accessible way to manage the mappings, but it has also caused such drawbacks as attention switching during demonstration on an object.

Although the user could successfully achieve his goal with the PBD application – it might not be the optimal solution for *simple* mappings that were implemented in the developed prototypes. Results show that compared to the GUI-based approach, PBD took slightly more time and effort from the user (section 6.6.2 "Questionnaire Results"). Partly this was caused by switching attention between the smartphone and demonstration on the object. Using a smartphone as a part of the PBD approach has its benefits: it is a familiar device for the user, it allows to easily correct the mappings, delete them and create new ones, any information about assigned grip gestures and objects can be stored there and is available for user's reference. However, once the user needs to interact with the object in the middle of his interaction with the smartphone the sequence becomes interruptive. There is a tradeoff between the familiar and non-interruptive interaction with the smartphone and the expressiveness of the PBD approach. The choice of the end-user programming strategy would depend on the specific task and its goals.

Benefits of the PBD approach

PBD allows direct interaction with an object.

One of the advantages of the PBD approach is the direct interaction with the object during programming. Users expressed that they liked the PBD application since it allowed them to try out the grip gesture before applying it. Users also mentioned that it was more natural to apply the grasp after assigning it to the object with PBD, since they could feel it beforehand. This could be useful if more complex grasps or motions are available: e.g., it could be hard to perform a correct grip gesture after choosing it from the GUI, but it is easier to repeat it if it was performed by the user during demonstration in PBD.

In the beginning of the study users were told that with the presented prototypes they can repurpose an object to be used only as a push button or a rotary knob. So during programming an object users only needed to indicate which area on the object (or which grip gesture) they would like to use, but they were not asked to perform (in case of

PBD can allow selecting a grip gesture and a way an object should be operated as a controller (as a knob, slider, etc.) in one step.

PBD) or select (in GUI-based application) any motion with the object. During the interview one of the users said that the systems (PBD and GUI) did not give him information that he should, for instance, rotate the object for changing some value. This user received the information about possible mappings from the investigator, but he emphasized that with PBD he could "grasp the object and rotate or slide it". This means that with PBD users could perform the grip gesture and the motion in one step, while in the GUI approach choosing an action could be an additional step or it would make the list of available mappings much longer, making the application more complex.

Machine learning can improve PBD approach and give users more flexibility.

Moreover, if the system could recognize any grip gesture performed by the user and learn it using AI methods, it would allow him to be more creative and personalize the interaction. We believe that PBD approach allows the user to be more expressive and map more individual and creative grip gestures or even motions.

6.8 Limitations

A limited set of implemented grip gestures is a limitation of the prototypes.

One of the limitations of the presented prototypes is the restricted set of grip gestures that were recognized by the system. This limitation is caused by the tracking technology. Other technical approaches apart from the vision-based technologies could be considered in the future work (see Section 7.3 "Future Work").

Using smartphone as an intermediate device has possibly limited the benefits of PBD approach.

Another factor, that could be considered a limitation, is the use of smartphone as an intermediate device during programming an object by demonstration. We selected this approach, since a smartphone is an accessible device that can be used for input and feedback from the system, i.e., for managing the mappings between objects and controllers. In future, our PBD approach combined with a smartphone could be compared to a PBD approach that does not require an additional programming device.

Chapter 7

Summary and Future Work

7.1 Summary and Contributions

In this work we investigated approaches for end-user programming of everyday objects using grip gestures. In particular, we focused on *Programming by Demonstration*. In order to understand the possibilities of this approach, we conducted a user study and compared PBD to another common end-user programming strategy – programming via a *Graphical User Interface*.

Users preferred to apply a grip gesture to different areas on an object compared to using a different number of fingers in a grip.

Programming everyday objects using grip gestures has not been covered in conventional research, therefore, before developing prototypes for the user study, we conducted a preliminary survey study to better understand which grip gestures are preferred by users. As a result, we found out that if users were asked to assign multiple controllers to an object of a cylinder shape, they prefer to use different areas on the object compared to using a different number of fingers in a grip gesture. We used these findings in the implementation of software prototypes.

We implemented two software prototypes representing a PBD and a GUI-based end-user programming strategy, that

During the study users assigned TV and Lights controllers to objects using PBD and GUI prototypes.

allowed users to assign mappings for TV and Lights control to everyday objects using a set of grip gestures. We conducted a user study and performed a qualitative evaluation of the developed prototypes by observation, questionnaires and informal interviews.

PBD and GUI prototypes performed equally well regarding usability characteristics.

We did not see a statistically significant preference of one approach over another, but we observed individual preferences. Users found both applications simple to use, understandable and easy to learn. Participants also stated that they were able to express their intentions and achieve their goals with both applications. Users expressed that they could effortlessly undo their actions and make corrections in mappings using both prototypes.

Projecting feedback directly on the object could improve the interaction.

Users were able to successfully interpret the visualization of previously assigned mappings in the PBD and the GUI prototypes, however, a possible improvement, suggested by users, would be to use a *projection on the object*. Projecting signifiers that represent controllers assigned to an object could reduce the cognitive load of remembering mappings and users would not need to use a smartphone as a reference.

Voice control could be an alternative to using smartphone as an intermediate device in PBD.

Including a smartphone as an intermediate device in the PBD approach was motivated by the fact that it provides an easy way for users to manage the mappings – users could create, correct and delete mappings through the application and receive a feedback on the smartphone screen. However, a drawback of this design is switching attention between the smartphone and demonstration on the object interruptive. In future, alternatives such as voice control or projection on an object could be used to allow users to focus on direct interaction with an object.

Participants expressed that the GUI-based approach was more familiar to them and, therefore, allowed to complete the tasks faster. However, with the PBD prototype users had an opportunity to try out the grip gesture before assigning it, and they found it more natural to apply the grip gesture after demonstrating it on an object, compared to selecting it from the GUI menu. Moreover, users expressed that the benefits of programming by demonstration would

be more valuable if *more complex gestures* with objects are available for mappings. For instance, if users could apply grip gestures with different angles, different size of the touch surface or even perform motions with objects, the list of possible mappings could be hard to be presented in a GUI. PBD, on the other hand, could use machine learning algorithms and learn any gesture demonstrated by users.

Machine learning methods could make PBD approach more expressive.

Another contribution of this thesis is the formulated *recommendations* for designing systems that implement end-user strategies for programming everyday objects, where we summarized the findings of the study.

7.2 Recommendations for Designing End-User Strategies for Programming Everyday Objects as Controllers

Based on the results of the conducted experiments, we created a list of recommendations for designing systems that enable end-user programming of everyday objects as controllers.

- *Use a GUI-based approach for simple tasks.*

  Programming an object via a GUI interface of a smartphone application is a satisfactory approach for simple mappings. If the task is to repurpose an object or areas of an object as a push button or a rotary knob, and the interaction does not include complex motions with multiple degrees of freedom, users preferred a GUI-based programming strategy. One of the supporting arguments is that interaction with a smartphone application is familiar to users, compared to, for instance, Programming by Demonstration. Moreover, if only a few (up to three or four in our experiment) areas of an object are available for mappings, users do not experience any problems to find corresponding areas on the real object, thus there is no need to demonstrate it.

- *Use Programming by Demonstration when more degrees of freedom for interacting with an object are available.*

  Compared to the GUI-based strategy, Programming by Demonstration is beneficial in scenarios where users can apply more complex interactions with objects. For instance, users can apply motions (which could be hard to visualize in a GUI) and create an individual gesture vocabulary, that is intuitive for them and appropriate for a particular task. PBD can give users more freedom in expressing themselves and make the system more flexible and personalized. PBD encourages users to experiment as they get to "try out" the gesture before applying it.

- *Present feedback on the object.*

  When users assign controllers to an object, they often have difficulties in remembering the mappings, as an object is usually not intuitively associated with a controlled system. This becomes especially noticeable if several controllers are mapped to an object with different grip gestures. Based on our users' feedback, we suggest that a projection on the object representing which areas are associated with which controllers will improve the experience from the interaction. This recommendation applies to both, the PBD and the GUI-based programming approaches.

- *In case of a PBD approach, present visual clues on the object indicating areas available for programming.*

  If the system is able recognize a limited number of interactions with an object, e.g., a set of areas on an object available for programming, and PBD approach is used, it is helpful for a user if this areas are highlighted by a projection.

- *Inform users about controllers that are currently assigned to an object.*

  It is important to provide users with the information about the current state of the object's mappings, i.e., which controllers and grip gestures are assigned to an object. It should be presented to users before they choose or demonstrate the gesture for a new mapping

in order to avoid accidental overwriting of the existing mappings or unnecessary pop up dialogs informing users that the area is already in use. The information about controllers and grip gestures assigned to an object can be presented in a smartphone application with a respective visualization, or, alternatively, by projecting signifiers on an object.

- *Avoid forcing users to switch attention during the interaction.*

  One of the benefits of using the GUI-based approach for programming everyday objects is that users interact only with a smartphone and do not need to switch their attention to other activities, such as demonstration on a real object. This should be considered in designing systems that use PBD as an end-user programming strategy: if a distraction is caused by switching from and to a smartphone, PBD can be combined with e.g., speech control for activating a controller, since it allows to keep the focus on an object.

- *Objects without possible state indicators are easier to be programmed as controllers.*

  We found it easier to program everyday objects that do not have possible state indicators. For instance, if users consider a handle of a mug such an indicator, and the controlled property (e.g., volume) has been changed from another source (e.g., from the original device), it will lead to the state inconsistency, as physical objects cannot be actuated.

- *The number of steps in a programming sequence can be reduced if users do not need to indicate minimum and maximum values of a property.*

  This recommendation is related to the previous one. If users take objects with possible state indicators, they could enforce the minimum and maximum values of a property. Although, there could be cases when this functionality is required, it will add additional steps to the programming sequence. When possible, use natural mappings, such as rotating an object clockwise to increase a value and counterclockwise to decrease it (as with a physical rotary

knob) and update the object position with a new value of a property if it was changed through another source.

- *Enable users to easily correct mappings.*

  As for any interactive system, a system providing programming everyday objects should enable users to easily make corrections. Since there could be approaches that allow assigning controllers to objects without an intermediate device (such as a smartphone), but use, for instance, voice control, a special vocabulary for making corrections might be necessary.

This concludes our list of recommendations for systems that enable programming everyday objects using grip gestures. Our findings intended to lay a foundation for future work in this domain.

7.3 Future Work

In future work, the PBD approach for programming everyday objects can be investigated further. According to the findings of the conducted user study, we expect that PBD will have more benefits in scenarios where more complex gestures and movements with objects are available. Moreover, machine learning methods could be implemented to enable users to teach the system any free gesture or grasp applied to an object. The examples of more complex gestures could be holding an object and moving it in a free space, or sliding a finger along the edge of an object. Users could throw an object or pass it to each other; also deformations of object's shape, such as squeezing or blowing into it, could also be used for mappings. PBD can give users an opportunity to be creative in creating new interactions. These interactions with everyday objects can be applied in various domains, not only in domestic environment for controlling digital devices as was demonstrated in this thesis, but also for controlling video game actions or composing music or other forms of digital art.

PBD might have more benefits as a programming strategy if more complex gestures with objects are available for mappings.

Other tracking technologies need to be explored.

More complex interactions with objects might require different technologies for its recognition. A vision-based approach might not always be the most reliable way for tracking interactions with objects as, firstly, occlusion can affect the tracking, and, secondly, it does not always allow exploiting physical affordances of objects. For instance, it could be hard to distinguish between a touch on a surface and a push. Another technical solution (that could be combined with a vision-based approach) includes using sensors: an accelerometer, capacitive sensor, pressure sensor, etc. Alternatively, electrical activity of the muscles could be used for identifying a grip gesture (as presented in Myo armband[1]), or resonance frequency of the object can be measured to define where on the object the touch has been performed (Ono et al. [2013]).

A PBD approach can be combined with voice control to avoid switching attention from the interaction with an object.

Although using a smartphone in combination with PBD approach can be a reasonable solution, as it provides users with an easy way to manage mappings, switching attention from the smartphone screen to the object during demonstration can be distracting and interrupting. Therefore, in future work alternatives, such as speech control or projection on the object could be investigated, as they do not require a user to switch attention from the object.

Information about the assigned mappings could be projected on objects to reduce the cognitive load of remembering the mappings.

Another factor, that we found, might improve the experience of programming and using an everyday object as a controller, is a projection of the assigned mappings on an object. Further research needs to be done to see how to visualize and present those signifiers, as it depends on the complexity of the gestures that are mapped to the object. For instance, if only three areas on the object are available for mappings — a projection could highlight the repurposed area on the object and present an icon of the assigned controller. However, if gestures with more degrees of freedom are available, visualizing the information about the gesture can be challenging. Projecting feedback on an object could help users to remember if and how an object was repurposed, and it can be applied independently from the type of the implemented end-user programming approach.

[1] https://www.thalmic.com/en/myo/

Appendix A

Appendix for the Preliminary Study: a Survey on Grasping Objects of a Cylinder Shape

This Appendix contains description of tasks and questions from the preliminary survey on grasping objects of a cylinder shape.

Q1. What is your age?

____

Q2. What is your gender?

- Female
- Male

TASK 1

Now please take a glass, try to grasp it from different sides or using different number of fingers, rotate it as a knob and try to figure out which grasp you would use, for example, to control the volume of your TV. Then go to the next page.

After experimenting with a physical glass please select the grasp below that corresponds to your choice of assigning volume control. If none of the pictures reflects your choice, please describe your suggestion in the comments box below.

Q3. Please select the letter corresponding to your choice for task 1 (grasp for controlling the volume):

- A
- B
- C
- D
- E
- F
- G
- H
- Any of the above
- Other (please specify)

TASK 2

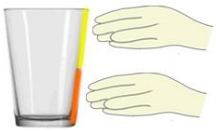
Now you need to use 2 grasps, for example, for controlling volume and screen brightness of your TV. Both functions should be assigned to one object (glass).

First take a glass and think which grasp you would use to control volume and which grasp you would use to control screen brightness then go to the next page.

After experimenting with a physical glass please select the grasps below that correspond to your choice of assigning volume and screen brightness. If none of the pictures reflects your choice, please describe your suggestion in the comments box below.

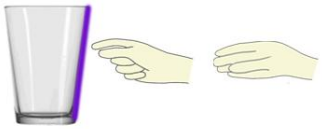
Please note that in each case grasps for volume and brightness are interchangeable (e.g. in case A top grasp can be brightness and side grasp can be volume). Given mappings are examples.

A. Grasp the glass from the top for volume control, and grasp it from the side for brightness control



B. Grasp the glass from the top side for volume control, and grasp it from the bottom side for brightness control

C. Grasp the glass from the top with 2 fingers for volume control, and grasp it with 5 fingers for brightness control



D. Grasp the glass from any area on the side with 2 fingers for volume control, and grasp it with 5 fingers for brightness control

Q4. Please select the letter corresponding to your choice for task 2 (grasps for controlling volume and screen brightness):

- A
- B
- C
- D
- Other (please specify)

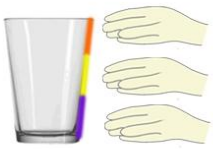
TASK 3

In this task you will assign 3 grasps to the object. In addition to volume and screen brightness control, you need to assign 3rd control - video navigation (so that you can rotate the object as a knob to fast forward or rewind the video). Please again experiment with your glass before going to the next page.

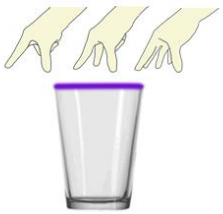
After experimenting with a physical glass please select the grasps below that correspond to your choice of assigning volume, screen brightness and video navigation. If none of the pictures reflects your choice, please describe your suggestion in the comments box below.

Please note that in each case grasps for volume, brightness and video navigation controls are interchangeable. Given mappings are examples.

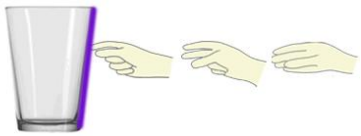
A. Grasp the glass from the top for volume control, grasp it from the side (top part) for brightness control, and from the bottom side for video navigation



B. Grasp the glass from the top side for volume control, grasp it from the middle side for brightness control, and from the bottom side for video navigation



C. Grasp the glass from the top with 2 fingers for volume control, 3 fingers for brightness control, and 5 fingers for video navigation



D. Grasp the glass from any area on the side with 2 fingers for volume control, 3 fingers for brightness control and 5 fingers for video navigation

Q5. Please select the letter corresponding to your choice for task 3 (grasps for controlling volume, screen brightness and video navigation):

- A
- B
- C
- D
- Other (please specify)

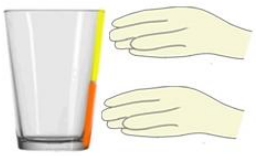
TASK 4

Imagine that you are editing your home video and you need a tool for video navigation with two levels of granularity - fine and coarse (e.g. going 1 frame per step and 10 frames per step). And this is what you want to assign to your object - so that when you use one grasp and rotate the object as a knob, navigation speed is 1 frame per step and when you use another grasp - navigation speed is 10 frames per step). Please take your glass and try different grasps for this task, then go to the next page.

After experimenting with a physical glass please select the grasps below that correspond to your choice of assigning fine and coarse video navigation. If none of the pictures reflects your choice, please describe your suggestion in the comments box below.

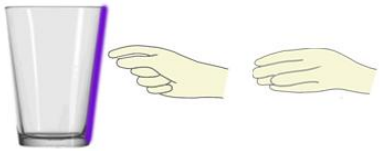
Please note that in each case grasps for fine and coarse navigation are interchangeable. Given mappings are examples.

A. Grasp the glass from the top for fine navigation and grasp it from the side for coarse navigation



B. Grasp the glass from the top side for fine navigation, and grasp it from the bottom side for coarse navigation

C. Grasp the glass from the top with 2 fingers for fine navigation, and grasp it with 5 fingers for coarse navigation



D. Grasp the glass from any area on the side with 2 fingers for fine navigation, and grasp it with 5 fingers for coarse navigation

Q6. Please select the letter corresponding to your choice for task 4 (grasps for fine and coarse video navigation):

- A
- B
- C
- D
- Other (please specify)

TASK 5

In this task we will ask you to choose an object and a grasp (gesture) for switching a device (e.g. TV) on and off.

Please choose if you would prefer to tap on top of the glass or to press the top of the stapler. You can suggest your own object and interaction (tap, press, etc.) in the comments box.

A. Tap on top of the glass to switch the TV on or off

B. Press on top of the stapler to switch the TV on or off

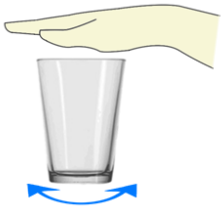
Q7. Please select the letter corresponding to your choice for task 5 (on/off switch with a tap or press):

- A
- B
- Other (please specify)

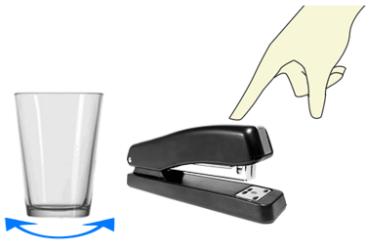
TASK 6

For this last task we will ask you again to choose the grasp (gesture) and the object for switching the device on and off, but in this case imagine that you have already assigned volume control to your object.

Please choose if you would assign tap on top of the same glass (as you are already using for volume control) for turning TV on and off or if you would take another object, which you can press, such as a stapler and assign the on/off functionality to it.



A. Rotate the glass for volume control with any preferred grasp and tap on top of the same glass to switch the TV on or off



B. Rotate the glass for volume control with any preferred grasp and press on top of the stapler to switch the TV on or off

Q8. Please select the letter corresponding to your choice for task 6 (on/off switch with a tap or press if volume has already been assigned to the glass):

- A
- B
- Other (please specify)

Appendix B

Appendix for the Interactive User Study

This Appendix contains full texts of Questionnaire A and Questionnaire B from the interactive user study.

Questionnaire A

Participant's ID____

1. The features of the system are comprehensive for me

strongly disagree	disagree	neither	agree	strongly agree
○	○	○	○	○

If something was not clear, please specify:

2. I found the sequence of actions reasonable for achieving my goal (from choosing the device to getting a confirmation of assigned connection)

strongly disagree	disagree	neither	agree	strongly agree
○	○	○	○	○

In case you found the sequence of actions rather not reasonable, please specify why:

3. I found GUI-based/PBD approach for programming everyday objects unnecessarily complex

strongly disagree	disagree	neither	agree	strongly agree
○	○	○	○	○

In case you found the system rather complex, please specify why:

4. I found the system with GUI-based/PBD approach for programming everyday objects easy to use

strongly disagree	disagree	neither	agree	strongly agree
○	○	○	○	○

In case you did not think the system was easy to use, please specify why:

5. I would imagine that most people would learn to use the system with GUI-based programming/PBD strategy very quickly

strongly disagree	disagree	neither	agree	strongly agree
○	○	○	○	○

6. I needed to learn a lot of things before I could get going with the system

strongly disagree	disagree	neither	agree	strongly agree
○	○	○	○	○

7. I felt very confident using this this system

strongly disagree	disagree	neither	agree	strongly agree
○	○	○	○	○

8. The system allowed me to express my intentions

strongly disagree	disagree	neither	agree	strongly agree
○	○	○	○	○

9. I found it easy to assign binary controls (push button) to an object using this system

	strongly disagree	disagree	neither	agree	strongly agree
TV Power	○	○	○	○	○
TV Next /Previous Channel	○	○	○	○	○
TV Favorite Channel	○	○	○	○	○
Lights ON/OFF	○	○	○	○	○
Lights Favorite Theme	○	○	○	○	○

If you found it rather hard to assign binary controls, please specify why

10. I found it easy to assign continuous controls (rotary knob) to an object using this system

*in this question,please, evaluate only whether it was easy to connect a grasp to a controller. Do not include the evaluation of mapping the minimum and maximum values

	strongly disagree	**disagree**	**neither**	**agree**	**strongly agree**
TV Volume	○	○	○	○	○
TV Screen Brightness	○	○	○	○	○
Brightness of the Lights	○	○	○	○	○
Color of the Lights	○	○	○	○	○

If you found it rather hard to assign continuous controls, please specify why

11. I found it easy to make corrections for the mappings between objects and controls using this system (here, please evaluate whether the system allowed you to change mappings without deleting a connection, e.g. going back to select/demonstrate a different grasp)

strongly disagree	**disagree**	**neither**	**agree**	**strongly agree**
○	○	○	○	○

If you found it rather hard to make corrections, please specify why

12. The visualization of previously assigned grasps (intermediate step during programming an object) was clear and helpful in achieving my goal

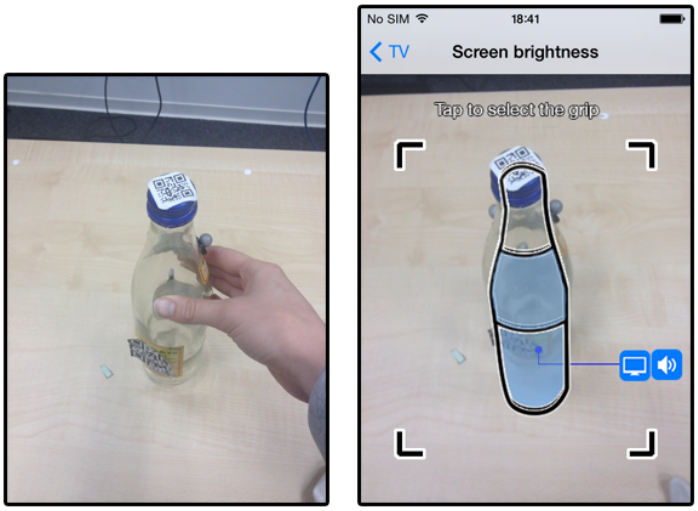
strongly disagree	disagree	neither	agree	strongly agree
○	○	○	○	○

If you found the visualization unclear or inconvenient please specify why

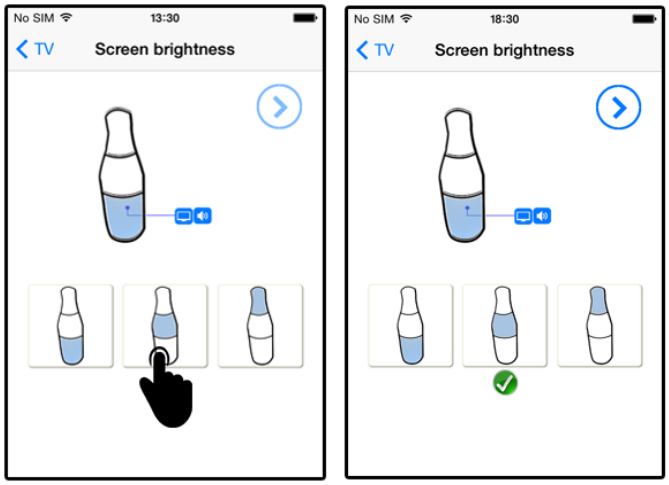
Questionnaire B

Participant's ID:_____

1. Please specify which visualization of chosen grasp and previously assigned grasps you preferred:



a)



b)

   a. Programming by Demonstration prototype
   b. GUI prototype
   c. I found both visualizations equal

*Could you think of **other** preferred visualizations or **improvements** to the suggested ones?

2. It was easier to **identify available grasps** with the GUI-based prototype than with the PBD prototype
   *see screenshots above

strongly disagree	disagree	neither	agree	strongly agree
○	○	○	○	○

3. I found it more **natural** to **apply** the grasp after programming it with PBD application than with the GUI approach.

strongly disagree	disagree	neither	agree	strongly agree
○	○	○	○	○

4. I found it **harder to demonstrate** the grasp in the PBD approach compared to **picking it from the options** on the screen in the GUI prototype.

strongly disagree	disagree	neither	agree	strongly agree
○	○	○	○	○

If you found PBD rather hard please specify why

5. I found it more **time-consuming to demonstrate** the grasp in the PBD approach compared to **picking it from the options** on the screen in the GUI prototype.

strongly disagree	disagree	neither	agree	strongly agree
○	○	○	○	○

If you found PBD more time consuming please specify why

6. I found **switching my attention** between the smartphone and interacting with the object for demonstration in the PBD application **interruptive**

strongly disagree	disagree	neither	agree	strongly agree
○	○	○	○	○

If you found switching rather interruptive, please specify why

7. During this user study you tried two prototypes which allowed you to map controllers to 4 areas of the bottle and 3 areas of the chocolate bar. Now imagine that you have more options of how the grasp can be applied (e.g. angle of grasp, size of the contact surface, number of fingers (with 2 fingers or with 5)), etc.

a. Do you think it is feasible to assign those mappings with the Programming by Demonstration approach?

strongly disagree	disagree	neither	agree	strongly agree
○	○	○	○	○

b. Do you think it is feasible to assign those mappings with the GUI-based approach?

strongly disagree	disagree	neither	agree	strongly agree
○	○	○	○	○

c. Do you find one approach more appropriate compared to the other? If yes, please specify why

Bibliography

Kai-Yin Cheng, Rong-Hao Liang, Bing-Yu Chen, Rung-Huei Laing, and Sy-Yen Kuo. iCon: Utilizing Everyday Objects As Additional, Auxiliary and Instant Tabletop Controllers. In *Proc. CHI*. pp. 1155–1164, 2010.

Keywon Chung, Michael Shilman, Chris Merrill, and Hiroshi Ishii. OnObject: Gestural Play with Tagged Everyday Objects. In *Proc. UIST*. pp. 379–380, 2010.

Christian Corsten, Ignacio Avellino, Max Möllers, and Jan Borchers. Instant User Interfaces: Repurposing Everyday Objects as Input Devices. In *Proc. ITS*. pp. 71–80, 2013.

Allen Cypher, Daniel C. Halbert, David Kurlander, Henry Lieberman, David Maulsby, Brad A. Myers, and Alan Turransky, editors. *Watch What I Do: Programming by Demonstration*. MIT Press, 1993.

Thomas R. G. Green and Marian Petre. Usability Analysis of Visual Programming Environments: A "Cognitive Dimensions" Framework. In *Journal of Visual Languages and Computing*. pp. 131-157, 1996.

Martin Hachet, Arash Kian, Florent Berthaut, and Myriam Desainte-Catherine. Opportunistic Music. In *Proc. JVRC*. pp. 45–51, 2009.

Björn Hartmann. Authoring sensor-based interactions by demonstration with direct manipulation and pattern recognition. In *In Proc. CHI*. pp. 145-154, 2007.

Valentin Heun, Shunichi Kasahara, and Pattie Maes. Smarter Objects: Using AR Technology to Program Physical Objects and Their Interactions. In *Proc. CHI EA*. pp. 961–966, 2013.

Rodrigo de A. Maués and Simone Diniz Junqueira Barbosa. Keep doing what i just did: Automating smartphones by demonstration. In *Proc. MobileHCI*. pp. 295–303, 2013.

John Napier. The prehensile movements of human hand. In *The Bone and Joint Journal*. pp. 902–913, 1956.

Makoto Ono, Buntarou Shizuki, and Jiro Tanaka. Touch & Activate: Adding Interactivity to Existing Objects Using Active Acoustic Sensing. In *Proc. UIST*. pp. 31–40, 2013.

Duck Gun Park, Jin Kyung Kim, Jin Bong Sung, Jung Hwan Hwang, Chang Hee Hyung, and Sung Weon Kang. Tap: Touch-and-play. In *Proc. CHI*. pp. 677–680, 2006.

Benoît Penelle and Olivier Debeir. Multi-sensor data fusion for hand tracking using kinect and leap motion. In *Proc. VRIC*. pp. 22:1–22:7, 2014.

Hayes Raffle, Cati Vaucelle, Ruibing Wang, and Hiroshi Ishii. Jabberstamp: Embedding Sound and Voice in Traditional Drawings. In *Proc. IDC*. pp. 137–144, 2007.

Naomi B. Robbins and Richard M. Heiberger. Plotting Likert and Other Rating Scales. In *Proc. JSM*. pp. 1058–1066, 2011.

Kimiko Ryokai, Stefan Marti, and Hiroshi Ishii. I/O Brush: Drawing with Everyday Objects as Ink. In *Proc. CHI*. pp. 303–310, 2004.

Raphael Wimmer. Grasp sensing for human-computer interaction. In *Proc. TEI*. pp. 221–228, 2011.

Robert Xiao, Chris Harrison, and Scott E. Hudson. WorldKit: Rapid and Easy Creation of Ad-hoc Interactive Applications on Everyday Surfaces. In *Proc. CHI*. pp.879–888, 2013.

James E. Young, Kentaro Ishii, Takeo Igarashi, and Ehud Sharlin. User-centered Programming by Demonstration: Stylistic Elements of Behavior. In *Proc. IJCAI*. pp.3106–3110, 2013.

Haiyan Zhang and Björn Hartmann. Building upon Everyday Play. In *Proc. CHI EA*. pp. 2019–2024, 2007.

Index

abbrv, *see* abbreviation

data communication, 62–65

evaluation methods, 72
everyday objects as controllers, 7

findings, 52, 90
future work, 98–99

grasping gestures, 34

interview, 72, 88–89

limitations, 91–92

observation, 72, 74–76

participants, 48, 74
preliminary study, 47
preparation phase, 39
procedure, 48, 68
programming by demonstration, 26
prototyping, 43–45

related work, 7
results, 48–52, 74, 76, 81, 88

scenarios, 69
system architecture, 60–62

tasks, 70

user action sequences, 57

vicon tracking system, 55

Typeset February 25, 2015