Improve Classroom Interaction and Collaboration using i>clicker

by

Junhao Shi

B.Eng., Zhejiang University, 2011

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in

The Faculty of Graduate and Postdoctoral Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

October 2013

© Junhao Shi 2013
Abstract

The i>clicker student response system is used to answer multiple-choice questions in university classrooms across North America. We investigated how existing i>clicker remotes could be used to improve classroom interaction and collaboration by developing and using custom software applications, each targeted at a different aspect of classroom interaction, that augment basic i>clicker capability.

Java-based software was written to replace the vendor-provided driver for the i>clicker base station that controls initialization, starting voting, requesting votes, stopping voting, and updating the LCD display on the base station. **WebClicker** extends voting to commonly-used digital devices (cell phone, smart phone, tablet, and laptop) using a cloud-based architecture that forwards votes to a client application.

Three client applications were developed. Each connects to either the Java-based driver or WebClicker to obtain votes. These extended the power of the standard i>clicker software. It supports most existing features, such as multiple-choice questions, but additionally features per-group visualization of voting outcomes, state-specific interpretation of individual student’s votes, and other features not in the vendor-provided software. **Clic^in** provides additional pedagogical support so students can practice newly-obtained skills in the class. It embeds “gamelets” that have content-specific behavior that can be played individually, or by an entire class, or in parallel by groups to support concept demonstration, class-wide participation and group competition. Finally, **Selection Tool** allows students to control projected material in the classroom through slide navigation and content highlighting.

Two usability experiments were conducted. One investigated cognitive load when using an i>clicker remote to interact with a gamelet that illustrates binary search tree insertion. The remote was slower and more error-prone than a mouse-based interface, but the difference is probably acceptable in a classroom setting. Both interaction time and error-rate decreased as participants gained practice. A second experiment compared **Selection Tool** with a mouse for content highlighting. Again the i>clicker was slower and more error-prone than a mouse, and it was difficult to correctly highlight smaller targets, but the ability to use an i>clicker for this task shows promise.
Preface

All the research work in this thesis was carried out under the supervision of Dr. Kellogg S. Booth. Ethics approval for experimentation with human subjects was granted by UBC Behavior Research Ethics Board (BREB ID H11-01756).

I am the primary researcher of all work in this thesis. Scott Newson was a co-participant in the development of the i>clicker base station driver that is described in Chapter 2. He was also the main contributor to the initial version of Clic^in that is described in Chapter 5. Laurence Baclin participated in the determination of the communication protocol between the i>clicker base station and a PC. Orkhan Muradov co-developed WebClicker++ that is described in Chapter 3.
Table of Contents

Abstract ................................................................. ii
Preface ................................................................. iii
Table of Contents ......................................................... iv
List of Tables ............................................................. vii
List of Figures ........................................................... ix
Acknowledgements ......................................................... xi

1 Introduction ............................................................ 1
  1.1 Student Response Systems ........................................... 2
  1.2 Research Contributions ............................................. 4
  1.3 Outline of the Thesis .............................................. 5

2 Reverse Engineering the Base Station Driver ....................... 8
  2.1 Understanding the Communication Protocol ....................... 9
  2.2 Base Station Driver Development .................................. 16

3 WebClicker: From Remotes to the Cloud ........................ 18
  3.1 Client Devices Support ........................................... 18
  3.2 Managing the Vote Flow in the Cloud ............................ 21

4 Clicker++: Reproducing and Improving the i>clicker Software ... 24
  4.1 Basic Clicker++ Functionality ................................... 24
  4.2 Improved Histograms ............................................. 25
  4.3 Specifying Groups ............................................... 26
  4.4 Classroom Beta Testing ......................................... 27
# Table of Contents

5 Clic\^in: Motivation, Architecture and Design ................... 29  
5.1 Motivation ........................................... 29  
5.2 Architecture Overview .................................. 30  
5.3 Design Specification .................................... 31  

6 Interacting with a Gamlet .................................. 33  
6.1 Binary Search Tree Gamelet ............................... 33  
6.2 Interaction Modes ....................................... 33  
6.3 Splitter, Filter, and Thresholder ......................... 34  
6.4 Examples of Other Gamelets ................................ 36  

7 Cognitive Load in Gamelets .................................. 41  
7.1 Background ............................................ 42  
7.2 Participants ............................................. 45  
7.3 Apparatus ............................................... 46  
7.4 Task and Procedure ...................................... 46  
7.5 Hypotheses ............................................... 48  
7.6 Results and Discussion ................................... 49  
7.7 Conclusion .............................................. 58  

8 Selection Tool: Slide Navigation and Content Highlighting using a Remote .............................................. 60  
8.1 Motivation ............................................. 60  
8.2 Overview of the Selection Tool ............................ 61  
8.3 Design of the Highlighting Tool ........................... 63  

9 Performance Assessment of the Content Highlighting Tool .... 65  
9.1 Participants ............................................. 65  
9.2 Apparatus ............................................... 66  
9.3 Task and Procedure ...................................... 66  
9.4 Hypotheses ............................................... 67  
9.5 Results and Discussion ................................... 70  
9.6 Conclusion .............................................. 75  

10 Conclusion and Future Work ................................ 77  

Bibliography .................................................. 79  

Appendices .................................................... 83
# Table of Contents

A Communication Protocol Between i>clicker Base Station (Old) and PC 84
   A.1 Initializing Base Station ............................................. 85
   A.2 Starting Voting ...................................................... 87
   A.3 Requesting Vote .................................................... 88
   A.4 Stopping Voting ..................................................... 91
   A.5 Updating LCD ....................................................... 92

B Communication Protocol Between i>clicker Base Station (New) and PC 94
   B.1 Initializing Base Station ............................................. 95
   B.2 Starting Voting ...................................................... 101
   B.3 Requesting Vote .................................................... 102
   B.4 Stopping Voting ..................................................... 104
   B.5 Updating LCD ....................................................... 107

C Participant Survey: Comparison of the Cognitive Load of Different Gamelet Interaction Techniques ......................... 109

D Participant Survey: Comparison the Cognitive Load of Different Gamelet Interaction Technique (Follow-up Study) ............. 113

E Participant Survey: Comparing the Performance of Highlighting using i>clicker Remote and Mouse ........................................... 116
List of Tables

2.1 A BSR Y packet sent by the base station .................. 15
7.1 Break-down of results in each step .......................... 53
A.1 Packets for the old base station ......................... 84
A.2 PCC 1 - Set the frequency of the base station .......... 85
A.3 PCC 2 - Set instructor’s remote ID ....................... 86
A.4 PCC 3 ........................................ 86
A.5 BSA 3 ........................................ 86
A.6 PCC 4 ........................................ 87
A.7 BSA 4 ........................................ 87
A.8 PCC 5 ........................................ 87
A.9 BSA 5 ........................................ 88
A.10 PCC 6 ...................................... 88
A.11 BSA 6 ...................................... 88
A.12 PCC 7 - Request new votes from the base station . 89
A.13 BSA 7 ...................................... 89
A.14 BSR 7 - New votes from the base station ............... 90
A.15 PCC 8 ...................................... 91
A.16 BSA 8 ...................................... 91
A.17 PCC 9 ...................................... 91
A.18 BSA 9 ...................................... 92
A.19 BSR 9 - Vote count summary from base station ....... 92
A.20 PCC 10 - Set first line of base station LCD .......... 93
A.21 PCC 11 - Set second line of base station LCD ....... 93
B.1 Packets for the new base station ......................... 94
B.2 PCC 1 ........................................ 95
B.3 BSA 1 ........................................ 95
B.4 PCC 2 ........................................ 96
B.5 PCC 3 ........................................ 96
B.6 PCC 4 ........................................ 96
List of Tables

B.7  PCC 5 - Set instructor’s remote ID .......................... 97
B.8  PCC 6 ......................................................... 97
B.9  BSR 6 - Base station firmware and frequency ................... 98
B.10 BSR X ............................................................ 98
B.11 PCC 7 ............................................................ 98
B.12 PCC 8 ............................................................ 99
B.13 PCC 9 ............................................................ 99
B.14 BSR 9 - Summary of the previous voting session ............... 100
B.15 PCC 10 ........................................................... 100
B.16 PCC 11 ........................................................... 101
B.17 PCC 12 ........................................................... 101
B.18 PCC 13 ........................................................... 101
B.19 PCC 14 ........................................................... 102
B.20 PCC 15 ........................................................... 102
B.21 PCC 16 ........................................................... 102
B.22 BSR Y - First new vote from the base station .................... 103
B.23 BSR Z - Follow-up new vote from the base station ............ 104
B.24 PCC 17 ........................................................... 104
B.25 BSR 17 ........................................................... 105
B.26 PCC 18 ........................................................... 105
B.27 PCC 19 ........................................................... 105
B.28 PCC 20 ........................................................... 106
B.29 PCC 21 ........................................................... 106
B.30 BSR 21 - Vote count summary from base station ............... 107
B.31 PCC 22 - Set first line of base station LCD ..................... 108
B.32 PCC 23 - Set second line of base station LCD .................. 108
List of Figures

1.1 Components of the i>clicker SRS .......................... 4
1.2 Base station driver, WebCicker, and client applications ......... 7

3.1 Laptop Service interface .................................... 20
3.2 Overview of WebClicker architecture .......................... 23

4.1 Histogram by lab group ........................................ 27

5.1 Overview of Clic^in architecture .............................. 32

6.1 A gamelet for concept demonstration .......................... 34
6.2 A gamelet for class-wide participation ........................ 35
6.3 A gamelet for group competition .............................. 35
6.4 Lifetime of a vote in a concept demonstration scenario .......... 37
6.5 Lifetime of a vote in a class-wide participation scenario ...... 37
6.6 Lifetime of a vote in a group competition scenario .......... 38
6.7 A linked list gamelet ........................................ 39
6.8 Procedure of linked list insertion .............................. 40

7.1 A slide showing a multiple-choice question .................... 42
7.2 Graphics interface of a BST gamelet .......................... 44
7.3 Error rate for each interaction technique and step combination ... 51
7.4 Interaction time for each interaction technique/step combination . . . . . . . . . . . . . . . . . . . . . . . . . 54
7.5 Error rate from the first tree to the last (fifth) tree ........... 55
7.6 Interaction time from the first tree to the last (fifth) tree ...... 56
7.7 Error rate from the first tree to the last (sixteenth) tree ....... 56
7.8 Interaction time from the first tree to the last (sixteenth) tree 57

8.1 A slide that has a mistake ...................................... 62
8.2 Content highlighting process .................................. 64

9.1 Examples of unsuccessful highlighting trials .................... 68
9.2 Part of the highlighted area inside the target .................. 69
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.3</td>
<td>Error rate for each device and size combination</td>
<td>71</td>
</tr>
<tr>
<td>9.4</td>
<td>Highlighting time for each device and size combination</td>
<td>72</td>
</tr>
<tr>
<td>9.5</td>
<td>Interaction time for each experience level and size combination</td>
<td>73</td>
</tr>
<tr>
<td>9.6</td>
<td>Coverage for different target sizes</td>
<td>74</td>
</tr>
<tr>
<td>9.7</td>
<td>Per-click time from the first to sixth click</td>
<td>75</td>
</tr>
</tbody>
</table>
Acknowledgements

First, I would like to thank my supervisor, Dr. Kellogg S. Booth, for his support, inspiration and guidance during my two-year M.Sc. program. Despite his busy schedule, he always tried his best to discuss the research project with me and offer me advice. I was always fascinated by his wide knowledge base, his unique perspective, and his problem-solving skills. Also, I would like to thank Dr. Romeo Chua, from the School of Kinesiology, for his time and effort as my second reader. The feedback he provided was extremely valuable.

I shall never forget my parents in any circumstance. Despite that they are living at the other side of the Ocean, their understanding, confidence, encouragement and support is essential to me. I am proud of them. I am lucky enough to have so many good friends with me, and they are also an important part of my life here in Vancouver.

The research reported in this thesis was supported by the Network Centers of Excellence Program (NCE) through the Graphics, Animation, and New Media (GRAND) network, by the Natural Sciences and Engineering Research Council of Canada (NSERC) through the Discovery Grant program, and by the Canada Foundation for Innovation (CFI), The University of British Columbia, and the Institute for Computing, Information and Cognitive Systems (ICICS) that provided research infrastructure. I would also like to thank all the participants for their precious time and valuable feedback.
Chapter 1

Introduction

In this thesis we explore how to improve classroom interaction and collaboration using the existing commercial i>clicker [5] hardware. These are a form of interactive student response system (SRS) that have recently been introduced into classroom lectures as one of many efforts to innovate the classroom experience.

Lectures have been for many years a popular form of educational experience at many universities. Most of the time, an instructor presents material related to the course, while students listen to the instructor. Once in a while, students raise their hands and ask questions when they are confused or need further clarification. Due to time constraints, however, communication between an instructor and students is often limited, especially for classes having large enrollments.

It can be important for an instructor to know whether students understand the material being presented so that he/she can better control the pace of the lecture. Yet this information is not available if the class as a whole cannot provide useful input. Additionally, a lecture encouraging interaction and collaboration would improve sense of involvement and participation among students. Thus, it is important to improve interaction and collaboration from the perspective of both instructors and students.

One way to achieve this goal is to adopt a student response system (SRS), which is very popular nowadays in some universities across North America. At The University of British Columbia the i>clicker student response system has been adopted and deployed in most classrooms and a number of courses use i>clickers during lectures. There are a few standard ways in which i>clickers are used, but all have somewhat limited interactivity and most are based on simply polling students opinion or knowledge by multiple-choice voting.

In our research, we looked for opportunities to develop more interactive applications based on the existing i>clicker system infrastructure. Deploying these applications does not require extra hardware, which makes it easier for instructors to adopt them in a real class. All that is required is the new software that we have developed.

After implementing new and largely platform-independent driver software for the i>clicker hardware, we developed a set of applications that used the driver to support in-class activities that engaged students by letting them interact with ma-
1.1 Student Response Systems

Student Response Systems (SRS) have been in existence for a few decades [27]. They are widely used in classrooms to allow students to provide immediate feedback so that both instructors and students can assess how well students understand the course material. Although there are quite a few commercial products in the market [2][6][3], many of them take a similar approach. Every student owns a remote that can transmit to a base station wirelessly. Students use their remotes to “vote” by pressing one of the keys on the remote to indicate their answers to questions posed by an instructor or to register their preferences or opinions when presented with multiple options. Connected to a classroom computer via a USB cable, the base station sends all the votes it receives from remotes to the computer. Software running on the computer then processes the votes and logs them, and optionally displays them, most commonly using either a histogram or a pie chart to summarize responses from the students.

There has been quite a lot discussion in the research community that suggests that use of a SRS has positive effects on the learning process. Many researchers found that students who used a SRS in class showed higher levels of attendance,
1.1. Student Response Systems

participation, engagement, and motivation [25][39] [31][22][24]. Additionally, it was shown that using a SRS helped students to better understand the course material in lectures [37][31][18]. However, researchers have not reached a consensus regarding whether using a SRS is helpful in terms of improving long-term learning. For example, some studies did find improvements of final exam scores among those who used SRS in class [25][37][22], but others did not [31][28][18].

There have been research articles pointing out that using a SRS is itself not enough to improve learning. Researchers have argued that many other factors come into play as well. For example, O’Donoghue and O’Steen believe that lecturers should develop approaches to using a SRS so that they could be used to align with students’ personalities or in a discipline-specific context [35]. Len discussed how different reward structures could motivate interaction when using SRS [30]. Trees and Jackson focused on how students’ characteristics and course design choices were related to SRS contribution and students’ involvement [40]. Crossgrove and Curran suggested that differences between courses targeting science majors and non-majors were important when considering using SRS [17].

Another research direction has investigated the effectiveness of applying SRSs in different topic areas or with different types of users. Researchers had examined SRS usage in various disciplines, such as general chemistry [25], biology [37], medical science [32], psychology [39], physiology [22], accounting [31], etc. Although SRS usage has mostly been studied in post-secondary institutions, Penuel et al. provided a survey describing how SRSs have been applied in elementary and secondary institutions [36]. Blood reported results of how SRS usage affected participation and learning of students with emotional and behavioral disorders [16].

The popularity of SRSs and the rich body of research that has already been conducted on them make us believe that SRSs could be a good platform on which to further develop classroom technology. Our own institution, The University of British Columbia (UBC), has adopted SRS (i>clicker) to support classroom interaction and collaboration. Many large classrooms in the campus are configured with i>clicker base stations. Student can buy an i>clicker remote at the UBC Bookstore. Figure 1.1 shows both the base station and the remotes currently used in UBC. These were also used in our research. Almost always, the i>clicker system is used for completing multiple-choice questions so that both the instructor and the students are able to evaluate student’s performance on the fly. Our research focuses on how to augment the basic multiple-choice-answer functionality of i>clickers to provide a variety of novel interaction paradigms in a classroom setting. We leave for others the on-going investigation of how well the basic functionality promotes learning. We hope that in future others will also look at the new functionality we have introduced.

3
1.2. Research Contributions

This thesis describes a set of research activities done under the framework of augmenting classroom technology using existing i>clicker hardware. The goal of the research is to fully explore the power of the i>clicker hardware and to expand classroom interaction and collaboration beyond simple multiple-choice questions. Below we summarize our research contributions by briefly describing each component of our work. Figure 1.2 is an overview of different components.

Our first step was to develop new base station driver software so that application developers could in the future create customized client applications based on i>clicker technology. Both the reverse engineering process that was followed and the communication protocols between remotes and the base station, and between the base station and computer, are described in detail. How the driver is used in client applications is also briefly described.

A later addition, WebClicker, was developed to support commonly used dig-
1.3 Outline of the Thesis

Digital devices in the voting process, such as cell phones, smartphones, tablets, and laptops. We designed and implemented a cloud-based architecture, as well as four web services, each targeted at one type of device. This generalizes the i>clicker hardware and provides a path for future migration to using more generally available hardware.

To test our ideas about how classroom engagement could be enhanced by introducing more interactivity into i>clicker usage, three client applications based on the i>clicker hardware and base station driver software were developed. These applications can be connected to either the base station driver or to the WebClicker infrastructure to obtain input from users. Clicker++ is a reimplemented version of i>clicker software with some design defects fixed. Clic^in is an application that allows students to practice their newly acquired knowledge in the classroom through new types of activities that we call “gamelets” that use i>clickers to guide the content using game-like mechanics. The Selection Tool provides an interaction paradigm that enables students to navigate through the slides being shown in the classroom and to highlight content on the slide, thus achieving more efficient communication among members in the class. All of the software has been tested in classroom situations, although the individual applications are still prototypes and have yet to be fully integrated into a package suitable for widespread deployment.

To assess some aspects of the new functionality, two lab studies were carried out. The results are presented, along with a description of the studies and their goals. The first study examined the cognitive load of interacting with a gamelet using an i>clicker remote. Gamelets are the main functional modules of Clic^in, one of the prototypes we developed. The second study compared the highlighting paradigm designed for the Selection Tool with a more traditional paradigm using a mouse to gain a better idea of an i>clicker remote’s performance in this task.

1.3 Outline of the Thesis

In Chapter 2 we describe the approach we used to build a base station driver, which enables developers to build client applications using existing i>clicker hardware. Chapter 3 describes WebClicker, a web application that allows students to vote using not only remote, but other commonly used digital devices as well. Chapters 4 through 9 talk about three client applications we designed to improve classroom interaction and collaboration. Chapter 4 presents Clicker++, a reproduction of the i>clicker software that fixes several design defects of the original version. Chapter 5 introduces Clic^in, a presentation application that supports a sequence of activities, which can be either static slides or interactive gamelets. Chapter 6 provides a detailed description of the gamelet concept, which makes different in-
1.3. Outline of the Thesis

teraction modes available, such as concept demonstration, class-wide participation, and group competition. Using a binary search tree gamelet as an example, Chapter 7 reports on an experiment investigating the cognitive load of interacting with a gamelet using an i>clicker remote. Chapter 8 demonstrates how the Selection Tool can be used to allow students to navigate slides and highlight content on a slide using an i>clicker remote, and Chapter 9 evaluates the interaction speed and error rate of the content highlighting tool, comparing it with a mouse. In Chapter 10 we summarize our research contributions and outline future work in this area.
1.3. Outline of the Thesis

Figure 1.2: Base station driver, WebClicker and client applications: Clicker++, Clic^in and Selection Tool.
Chapter 2

Reverse Engineering the Base Station Driver

One important reason for choosing i>clicker as the hardware platform for improved classroom interaction is that it has been adopted at UBC, which means that a high percentage of students own an i>clicker remote and are familiar with the technology, and that there is existing hardware support installed in most classrooms on campus. The advantages are two-fold. First, easy access to hardware in classrooms makes it convenient to test applications with real users because no special arrangements need to be made. Second, it means that new applications developed in the research can be deployed for general use in any classroom by any instructor without the need for any new hardware or software except for a simple web-based download of software to the instructor’s laptop. This reduces costs both for the university and for students.

Unfortunately, the standard software that is provided for i>clicker does not provide all of the functionality that we need to implement some of the new applications we designed. In order to achieve our research goals, a way had to be found to access and process votes collected by the base station in real-time to drive the applications, not just collect votes for subsequent off-line processing, which is what the standard i>clicker software does.

Prior research in our lab had built a prototype that modified the source code for the vendor-supplied software to support some of the additional functionality. That served as a proof of concept, but it was not extensible, both because the prototype used the “trick” of bypassing most of the actual vendor software by sending the low-level data to another process via a socket connection, but also because source code for subsequent versions of the vendor software that supported newer hardware were not available to us. So we had to first develop our own software driver on which to build the applications we had designed.

This chapter describes how an i>clicker remote communicates with a base station, how the communication protocol between the base station and a computer was determined through a reverse engineering process, and how the base station driver was then augmented, using that existing protocol, to provide support for advanced applications we had designed.
2.1 Understanding the Communication Protocol

We provide in the next subsections an overview of the basic operation of the hardware and a high-level description of how the communication protocols between the remotes and a base station and between a base station and a PC work, but not all of the details. A full description of the packet format for the old “black” base station is provided in Appendix A and Appendix B has a full description of the packet format for the new “white” base station.

Throughout this section, PC refers not only to the physical computer that connects to the base station, but also to the drivers and applications installed on the computer that communicate with the base station, and signifies either a Windows PC or an Apple Macintosh.

2.1.1 Hardware description for a base station

The base station is connected to a PC by a USB cable. The base station has an LCD screen, which can display a total of 32 characters, evenly split into two rows. The base station can either be in “accepting” or “not-accepting” mode. It processes students’ votes only when it is in “accepting” mode. However, instructor’s votes can be received in both modes.

At the hardware level, the base station is always receiving transmissions from every remote, but the firmware ignores any but those from the instructor’s remote unless it is in “accepting” mode. The base station “knows” which remote belongs to the instructor because software running on the PC establishes this as part of the initialization sequence when it connects to the base station.

2.1.2 Hardware description for a remote

A remote has six keys and three lights. Among the six keys, the first five are vote choices (A to E), and the last one is used to turn power on and off for the remote. The last key also has a special role in setting the channel frequency for a remote, which will be described later. All lights are off when the remote is turned off. When it is turned on, different lights use different colors to indicate different status.

The first light is labeled with “POWER”. It will light up in blue when the remote is turned on. Function of the first light when a user changes the channel frequency will be described later in this subsection.

The second light on a remote is labeled with “LOW BATTERY”. Usually it is off and it will blink red as an indicator of low battery.

The third light on a remote is labeled with “VOTE STATUS”. It indicates whether a vote has been successfully received by the base station. Once the base
2.1. Understanding the Communication Protocol

The base station receives the vote, it will send an acknowledgement back. A long blink in green of the third light indicating the acknowledgement is received by the base station. Otherwise the user will see four short blinks in red, indicating that no acknowledgement is received. The third light is also used when the user changes the remote’s channel frequency, as described later.

Failure to receive acknowledgement can occur because the base station is not connected to a PC, or because the base station and the remote do not share the same channel frequency, or because the base station is in “not-accepting” mode, or because there is a collision with another remote that is also transmitting at the same time. In all of these cases, no acknowledgement will be generated by the base station and the remote will eventually time out.

A user changes the channel frequency by pressing and holding the power key on the remote for a few seconds until the “POWER” light begins blinking. The user then enters the two-key frequency code corresponding to one of the 16 two-letter combinations of the letter A-D. The “POWER” light will keep blinking in blue until the two keys have been pressed. It quits blinking only if the new frequency code is successful in sending the remote’s ID and a base station sharing the same channel frequency sends an acknowledgement back on the second frequency. Otherwise the “POWER” light keeps blinking. After the first frequency code is entered, the third light will light up and stay in orange. After the second frequency code is entered, it will blink once in green if the acknowledgement is received. Otherwise four short blinks in red will be witnessed.

2.1.3 Communication between Remotes and a Base Station

Communication between remotes and a base station takes place wirelessly. Each remote has a unique 32-bit remote ID that is comprised of a unique 24-bit ID followed by an 8-bit checksum that is the exclusive OR of the three 8-bit bytes that make up the 24-bit ID. For a remote whose ID is 0x058549C9, the checksum, which is represented by the last two characters 0xC9, is the exclusive OR of 0x05, 0x85 and 0x49, namely, 0xC9 = 0x05 \text{ XOR } 0x85 \text{ XOR } 0x49. Early versions of the i>clicker system used only the low-order 21 bits of the 24-bit ID, but with the same 8-bit checksum calculation.

The 32-bit ID is written in hexadecimal on a sticker that is attached to the back of the remote, and also on a second sticker that is inside the remote. The inner sticker is only accessible by removing the screws from the casing of the remote. Normally this is not done by a user, but for many remotes in the earlier shipments that were sold at the UBC Bookstore, the outer stickers had their ID numbers wear off, so it was helpful that there is another inner sticker that also has this information.

Even though the 32-bit remote ID itself could support error detection, the error
2.1. Understanding the Communication Protocol

detection is never carried out during the transmission process. When a remote sends a vote to the base station, it includes the first 24 bits of the ID and the key pressed, but not the checksum. The base station on the receiver side recovers the 32-bit ID by carrying out the exclusive OR operation on the 24-bit ID it received and appending it as the final 8 bits of a 32-bit ID. Thus, the base station has no way to determine whether the 24-bit ID is correct. Similarly, in the USB communication between the base station and the PC, only the first 24-bits of the ID and the key pressed is transmitted. Thus, there is no way to detect errors during the wired transmission as well.

The exclusive OR operation could help detecting errors in some cases, but not always. For example, both 0x058549 and 0x058448 produce the same checksum 0xC9, so there is no way to achieve error correction in this case. For example, if the base station receives a 32-bit ID 0x058349C9, and it found that the checksum computed from the first 24 bits is 0xCF, instead of 0xC9, the base station cannot tell which bits are incorrect because more than one 24-bit IDs can generate the same checksum, which is the case for 0x058549 and 0x05834F. It is also possible that the checksum itself is contaminated by one or more errors. These are well known deficiencies of exclusive OR when used as a checksum. More robust error detection and error correction codes could be employed [21] [38][26], but this would require a firmware change to the base station and the remotes, as well as a software change on the PC because current communication protocol only allows the base station to transmit the first 24 bits of a remote’s ID, not the 8 bits of the checksum.

Communication is asymmetric. Remotes send information to the base station in parallel with each other, using a single shared frequency. This means there is a possibility of collision when more than one remote transmits at the same time. The base station serially broadcasts its acknowledgements on a different frequency, so there is no possibility of collision for acknowledgements being sent to a remote. We were not able to determine what happens if there is a collision, but presumably this results in the base station receiving a corrupted packet and most probably it sends an acknowledgement that does not correspond to any of the remote IDs whose transmissions were part of the collision, resulting in all of the remotes timing out.

Based on the scant evidence we collected about how long it takes for a large number of close-to-simultaneous votes to be collected, we suspect that when a remote times out it tries to re-transmit some number of times before finally indicating an error if it is not successful, and that each time it re-transmits it first waits an increasingly longer time, somewhat like the “exponential backoff” protocol used in Ethernet [15] and various other communication schemes.

There are 16 pairs of frequencies that are used by the base station. Each pair is identified by two letters in the range A-D. During the change of channel frequency,
after two keys are pressed, the remote uses the two keys to determine the pair of frequencies it will use to transmit and receive from the base station. The remote then sends its ID to the base station using the selected transmit frequency and listens for an acknowledgement on the receiving frequency.

Efforts were made to understand how the wireless communication between a remote and a base station works. Previous work [23] managed to get the firmware from the remote, and we followed the same approach. After disassembling the remote, we found that the AVR in-system programming port was exposed on the printed circuit board (PCB) in six pins, which allowed us to read and reflash the memory. An AVRISP mkII [1] was used to dump the program memory content to file. Then Atmel Studio 6 [8] was used to convert the binary code to hex code and then decompile that to assembly code. We were only able to determine that when a key is pressed, different values are stored in register R16 indicating which key is pressed (0x02, 0x04, 0x08, 0x20, 0x10 for A-E, respectively, and 0x01 for the power key). Presumably each value corresponds to an input pin for a key. At some other point during execution, the first three bytes of the remote ID are stored in registers R16, R17 and R18, but we do not know how this information is combined and transmitted to the base station because we were not able to fully understand the assembly-level code in the firmware.

Despite not fully understanding the communication protocol between the remotes and the base station, we obtained enough information to proceed with our research through our reverse engineering exercise. Figuring out the protocol between the base station and the PC was easier because we had a USB sniffer, but the protocol was more complicated so there was more to figure out.

### 2.1.4 Communication between the Base Station and a PC

The communication protocol between a computer and a base station was obtained by reverse engineering. That is, we ran the i>clicker software in the way it was normally used in a classroom, and at the same time we ran a USB traffic sniffer software on the same computer to capture all of the incoming and outgoing packets exchanged between the base station and the computer. Whenever any event happened, such as the instructor starting or stopping a voting session or a student contributing a new vote, the corresponding packets were recorded and analyzed.

Both the new “white” i>clicker base station and the old “black” i>clicker base station hardware were used in the reverse engineering process. Only the protocol for the new “white” base station is presented in this chapter, but both were fully analyzed as part of the research and the augmented drivers that were later developed are compatible with both types of base station.

Version 6.1 of the i>clicker software was used throughout the reverse engi-
neering process for the new “white” base station, which had firmware version 4.5. For the old “black” base station, the i>clicker software was version 5.4.5 and the firmware version was 2.3. The “sniffing” software USBlyzer [14], version 1.4, was used to capture all packets. Other types of USB analyzers could have been used, but this software-only solution met our needs and avoided the need to purchase additional hardware.

All packets sent between the base station and the PC have 64 bytes. On Windows machines, however, when a PC sends a packet to the base station, one extra byte is added at the beginning of the packet, with a value of 0x00, making the packet 65-byte long.

The PC has full control of when to initialize the base station. It specifies the instructor’s remote ID, what pair of channel frequencies to use, when to start and stop voting, and what to display on the base station LCD screen. These are determined by commands that the PC sends to the base station. The base station simply interprets and carries out the commands.

To connect to a base station, a USB cable is attached to the USB port on the base station and to one of the USB ports on the PC, or to a USB hub that is connected to the PC. The PC is then responsible for specifying the instructor’s remote ID so that the base station can receive the instructor’s vote even in “not-accepting” mode. Until this is done, the base station will only accept commands from the PC via the USB connection; it will not accept votes from the instructor’s remote because it has not been provided with the proper ID so it cannot tell which votes come from the instructor.

### 2.1.5 Format and Protocol for Votes

All votes share the same format, no matter whether they come from the instructor’s remote or from students’ remotes. The base station is told the ID for the instructor’s remote by the PC, which it treats as special. The base station always receives and processes the votes that come wirelessly from every remote no matter whether the base station is in “accepting” or “not-accepting” mode. However, it will send all votes to the PC if it is in “accepting” mode, but it discards everything it sees except for votes from the instructor’s remote if it is in “not-accepting” mode.

When voting is turned on, it is the PC that actually separates students’ votes from the instructor’s votes, and it is up to the PC to decide how to interpret different choices implied by the instructor’s votes and by students’ votes. The vendor-supplied software has a fixed set of interpretations for votes from the instructor that is recognizes. The firmware in the base station does not know the interpretations. It passes the instructor’s votes on to the PC, which interprets them and sends back appropriate commands to the base station.
2.1. Understanding the Communication Protocol

Votes from students are also uninterpreted by the base station, and by the PC as well. They are simply logged by the i>clicker software for subsequent analysis, although the software does provide capabilities to display the number of votes or histograms of counts of the five possible vote values that have been received during the current voting session, as well as some other summary information about aggregate votes. But the software does not do any special that depends on the value of a particular student vote. Any distinctions are made off-line by evaluation or grading software that processes text files in .csv format that are produced by the i>clicker software. These record some aspects of the votes that were received, as well as timestamps and other statistical information such as the number of votes cast by each remote.

Starting voting moves the base station from “not-accepting” mode to “accepting” mode.

Once the base station receives a new vote, it sends a packet to the PC. The packet includes the key that was pressed (A-E), the first six characters of the hexadecimal remote ID, and the index of the vote in a 256-entry circular buffer (so indices range from 0x00 to 0xFF, and then go back to 0x00 again). Packets will stay in the PC’s I/O memory buffer, waiting to be read by programs running on the PC.

Stopping voting moves the base station to “not-accepting” mode. Additionally, the base station will send a summary packet that includes information such as how many votes have been collected in total, the instructor’s remote ID, and other status information.

2.1.6 Controlling the LCD Display on the Base Station

The PC decides what to show on the base station’s 32-character LCD screen, and when to update the content displayed. The base station displays the character string it receives from the PC. The vendor-supplied i>clicker software on the PC updates the LCD screen every second by sending the appropriate command and data.

2.1.7 Summary of Commands for the Base Station

In Appendix B we provide a full list of all the packet types sent to and from the base station. Here we only provide a partial description. For the ease of description, we classify packets according to both their roles and associated tasks. There are three types of roles. All the packets sent from the PC to the base station are called PC Command (PCC) packets. Acknowledgement packets sent by the base station are called Base Station Acknowledgement (BSA) packets. All non-acknowledgement packets sent by the base station are in the third group, Base Station Response (BSR) packets. Acknowledgement packets sent by the base station are called Base Station Acknowledgement (BSA) packets. All non-acknowledgement packets sent by the base station are in the third group, Base Station Response (BSR) packets.
2.1. Understanding the Communication Protocol

Packets. BSR packets usually include some useful information, for example, information about the base station itself, votes received, etc. Also, a packet is sent to complete one of the following five tasks: initializing the base station, starting voting, requesting votes, stopping voting, and updating the LCD display on the base station.

Because the number of commands is huge, the reader is referred to the appendices for details. Here, for the purpose of demonstration, we show what packet is transmitted when voting is going on (the packet type BSR Y in Appendix B, it is a BSR packet, and it is sent for requesting votes).

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>0x13</td>
<td>Choice1</td>
<td>ClickerID1</td>
</tr>
<tr>
<td>ClickerID1</td>
<td>VoteIndex1</td>
<td>0x00</td>
<td></td>
</tr>
</tbody>
</table>

... (6 rows, omitted, all being 0x00) ...

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>0x13</td>
<td>Choice2</td>
<td>ClickerID2</td>
</tr>
<tr>
<td>ClickerID2</td>
<td>VoteIndex2</td>
<td>0x00</td>
<td></td>
</tr>
</tbody>
</table>

... (6 rows, omitted, all being 0x00) ...

Table 2.1: A BSR Y packet sent by the base station.

As long as the base station has received new votes that have not been transmitted, it will send a packet in the above format to the PC. One packet contains two votes, the current vote and the previous vote (except for the first packet, which only contains one vote). Each vote contains an index value (indicated by the fields VoteIndex1 and VoteIndex2 in the table, respectively), which starts incrementing from 0x00 for each incoming vote, and it goes back to 0x00 when it reaches 0xFF, so that 0xFF is considered to be smaller than 0x00 even though numerically it is larger. The vote with a larger index value (in the circular sense just described) is the current vote, while the one with a smaller index value (in the circular sense) is the previous vote. Choice1 and Choice2 are the respective choices for the two votes, and ClickerID1 and ClickerID2 are the first six digits of the IDs for the remotes that sent the two votes.
2.2 Base Station Driver Development

With the full communication protocols at hand, it was straightforward to write driver code to manipulate the base station. For the purpose of modularization, a stand-alone driver was developed. The source code for the driver can also be used as a building block for any application that utilizes i>clicker hardware, either reusing some of its components or by attaching to a process that runs it through a socket connection.

Two platforms were considered, Microsoft Windows and Mac OS. These are the dominant systems used in classrooms nowadays. Java was chosen to build the driver (as well as other applications in this project) because of its cross-platform capability. We believe that our software could easily be adapted to a Linux platform if there is a need to do so.

We used the HIDAPI [9] library because it allows an application to easily interface with USB and Bluetooth HID-Class devices. The javahidapi [10] package is a JNI wrapper for the C/C++ HIDAPI library that provides a Java interface to work with these devices. Using javahidapi, a driver for the i>clicker base station was developed that works on both the Windows and Mac OS platforms.

Although packets being sent have different types of roles, each packet aims at completing one of the five tasks, as stated previously: initializing the base station, starting voting, requesting votes, stopping voting, and updating the LCD display on the base station. This provides a fairly clean interface for client applications. In Java, it is a class with one constructor and five additional public methods. The implementation of each method includes constructing packets and sending them to the device, receiving acknowledgment packets if there are any, and receiving response packets and parsing their content if there are any. Some commands sent to the base station receive no packets back. Other commands receive just an acknowledgement packet but others receive just a response packet. Some commands receive both an acknowledgement packet and then a response packet.

After sending a packet to the base station, the driver has to wait for the response or acknowledgment packet’s arrival before trying to read them. However, the lag from a command being sent to the base station to the moment when the acknowledgment or response packet arrives back at the PC differs greatly for different commands. This was determined by experimenting with the vendor-provided software.

The exact lag values were obtained by running the i>clicker software and the USBlyzer software application at the same time, recording the timestamp when a certain packet was sent and also when its acknowledgment or response packet(s) arrived, and then computing the differences of the timestamps. This process was repeated several times for each command sent by the PC and the results were av-
2.2. Base Station Driver Development

eraged. For several commands, the lag values were quite large (longer than one second). To make the driver run more efficiently, smaller values were used and tested, but this unfortunately led to non-functional code so we reverted to using the larger values.

Once appropriate lags were determined for each command type the driver software was ready to be used to develop applications for enhanced i>clicker usage in classroom.
Chapter 3

WebClicker: From Remotes to the Cloud

The base station driver described in the previous chapter only supports voting via regular clicker remotes. However, the pervasiveness of personal computing devices provides a new chance for classroom interaction and collaboration. Nowadays, most students own at least one of the following personal devices: a mobile phone (the majority of which are smart phones), a laptop, or a tablet. These devices are all very portable and are commonly brought with students to class. They provide significantly more powerful computational capabilities, as well as interactivity, compared to most i>clicker classroom remotes, which only have five keys. Here, we focus on the development of WebClicker, a cloud-based system that supports classroom activity via the personal devices such as those just listed. This chapter describes the structure of WebClicker and demonstrates how it can be used to provide input for various client applications.

3.1 Client Devices Support

To integrate personal devices into the classroom, they need to be able to communicate over a network to the software that is being used by an instructor. The wireless network at UBC seemed at first to be a good choice, however, due to security constraints, point-to-point connections through Wi-Fi have been disabled (connection requests are filtered by the campus network software and are not forwarded to clients that are connected via Wi-Fi). Therefore, an alternative communication mechanism was needed. Our solution was to use a web server external to the UBC network that could be set up to process all the incoming votes and feed the results back to Clicker++ (Chapter 4), or any other client application that runs on an instructor’s computer, such as Clic>in (Chapter 5) or Selection Tool (Chapter 8). Through a browser, a web service provides a lightweight approach for students to access the application using various client devices. Using a web service is also useful for the instructor because it centralizes all of the administrative information and vote session history in the cloud, which facilitate data management and retrieval,
3.1. Client Devices Support

especially if there are multiple instructors sharing a course.

We developed four different web services to support various devices. *Remote Service* uploads votes received by the base station to the cloud. *Laptop Service* runs a website that is designed for laptop users, while *Mobile Service* is built for smartphone and tablet users. Finally, *SMS Service*, which is built on top of Twilio[12], receives SMS messages sent by students and converts them to votes.

3.1.1 Remote Service

Remote Service deals with votes coming from clicker remotes. The basic idea is that after the base station receives a vote and sends it to a PC, a program in the PC then uploads the vote to the cloud. However, experimental results show that this solution leads to a significant delay, mainly because uploading one single vote immediately when it is received is expensive. A workaround is to buffer the votes, and send them all together at regular intervals.

The implementation of Remote Service uses the base station driver described in Chapter 3 to receive votes within a classroom setting and then forwards them to the cloud-based WebClicker server using standard networking protocols. Connections are established by the Remote Service client contacting the WebClicker server, which is seen as an out-going connection request by the local Wi-Fi network, so the connections are allowed to proceed. The additional software for Remote Service is straightforward. It uses the API provided by the base station driver to monitor the local classroom remotes and it uses the API of the WebClicker server to forward votes into the cloud where they are processed.

Only instructors have to use the Remote Service application, which they can download from the WebClicker website. Once installed, it operates like an extended version of the base station driver, with the actual management of classroom activities, such as voting, controlled by cloud-based software accessed using a normal web browser.

3.1.2 Laptop Service

Laptop Service allows students to vote during a class by connecting to a web page for voting on a website. It also provides support for registering in classes within the service and for managing student devices (e.g., multiple clicker remotes or cell phones). An instructor can upload course registration information so that only students who take a course can successfully register with the service.

The voting page contains an image of an i>clicker remote, with five keys to click (see Figure 3.1). Sound (which can be turned off) and LED light effects are added so that the voting page mimics a real clicker remote when it is used.
3.1. Client Devices Support

Feedback is provided every time a student clicks on a key on the simulated remote. This shows whether the student’s click has been successfully received, what the choice was, and the timestamp for the vote.

There is also a module which helps students to manage their devices. Students can add multiple remotes by specifying the remote ID for any additional remotes they might use, so that they need not remember which remote is registered for the class. They can also add a cell phone by specifying its phone number, enabling them to vote by sending text messages (via the SMS Service, described later).

Only students use the Laptop Service. It is accessed using a normal web browser through a URL provided by the instructor.

![Laptop Service interface](image)

Figure 3.1: Laptop Service interface.

3.1.3 Mobile Service

Mobile Service is the mobile interface for Web Service. It supports all of the functionalities provided by Laptop Service, but presents an interface that nicely fits the dimensions of mobile devices through the use of Kurogo [11], an open source mobile optimized middleware for developing mobile websites. Kurogo’s power lies in its ability to format the layout of the content based on the type of device initiating the web request. It also has a lot of built-in features that are standard for today’s mobile applications, such as database access, user access and...
3.2 Managing the Vote Flow in the Cloud

Amazon Web Service [7] was chosen for WebClicker because it is free and the quality provided is good enough for research purposes. The functionality of WebClicker is similar to that of the standard vendor-provided i>clicker application, with extensions to support new features that were added in Clicker++ (Chapter 4) and the other i>clicker applications we developed that use the base station driver.

When a new semester starts, an instructor uploads a list of user IDs of all the registered students to the WebClicker and tells students to sign on WebClicker. A user ID can be any unique ID that can identify a student. The user ID list is provided so that WebClicker will only allow students who registered in this course to sign up. Students sign up on WeClicker using either the Laptop Service or the

3.1.4 SMS Service

We realize that not everyone owns a smart phone. Support is therefore provided to regular cell phone users by establishing SMS Service, which is based on Twilio, a platform that enables phones and messaging to be embedded into web applications. By applying for a local cell phone number and associating it with an SMS request URL, anyone who sends a message to this number will initiate an HTTP request to the specified URL, which in our case is the WebClicker server. The request is processed by a Java Servlet on the WebClicker server.

Students only need to know the phone number to which they send their vote message. Besides, they need to register their own cell phone number with the WebClicker server prior to voting, via either the Laptop Service or the Mobile Service. To submit a vote, a student specifies the phone number, provides the choice (one character long with the value being one of ‘A’ to ‘E’, case insensitive), and then confirms by pressing a button such as “OK” or “Send”. The SMS Service takes more steps to submit a vote than required by a regular remote or by the Laptop Service or the Mobile Service, but it does permit a regular cell phone owner to participate in classroom activities without having to use a more specialized device. The process would take less time if the phone number is saved as a contact so that students do not need to specify it every time they vote.

3.2 Managing the Vote Flow in the Cloud

authenticati, session management, system log and administrator management, etc. All of these features easeed the process of development and management of the Mobile Service.

Students can access the Mobile Service with a browser application, which is installed on a wide variety of smart phones and tablets.
3.2. Managing the Vote Flow in the Cloud

Mobile Service by providing their user ID and a password, with which they can log on to WebClicker via either the Laptop Service or the Mobile Service in the future. After logging on to WebClicker, students can add and manage their remote ID or phone number.

For each course, WebClicker maintains a database that keeps a record of all students’ WebClicker registration information: their remote ID and their cell phone number (if any), as well their user ID if they access the Laptop Service or the Mobile Service.

All services listed above will access the database when they receive new votes. Votes are logged in the database, in addition to being sent to the client that is running. In the database a vote record contains the following information: user ID, timestamp when the vote is received, choice, service code (R for votes received via Remote Service, L for Laptop Service, M for Mobile Service, and S for SMS Service), and the device ID if any (phone number for regular cell phone and remote ID for a clicker remote). Currently client applications utilize the user ID, timestamp, and choice information. The service code and the device ID are not used in any of our existing applications, but these fields may be useful for future research studying users’ preferences about different devices.

See Figure 3.2 for an overview of the WebClicker system. Votes from any of the four services (Remote, Laptop, Mobile, and SMS) are collected in the cloud database and then fed to clients to which an instructor connects in the classroom.

Any client application that was originally designed to use the i>clicker base station driver (Chapter 2) can get the vote flow through WebClicker using the API developed for the driver. In the next few chapters, we will describe some of these client applications in detail.
Figure 3.2: Overview of WebClicker architecture.
Chapter 4

Clicker++: Reproducing and Improving the i>clicker Software

Using the base station driver and WebClicker, three client applications were developed to fully exploit the power of the i>clicker hardware: Clicker++, Clinc^in and the Selection Tool. These applications all improve different aspects of the classroom interaction and collaboration experience supported by the basic vendor-provided i>clicker software. They can be connected to either the base station driver or to WebClicker (Chapter 4) through a network socket, thus maintaining modularization. The driver or WebClicker simply acts as a vote collector that sends all received votes to the client application for processing.

Each vote contains three pieces of information: the remote ID, the key clicked, and a timestamp indicating when the driver received the vote (as mentioned in the previous Chapter, WebClicker also provides a service code and a device ID, but these are currently not used by client applications). It is up to the client application to decide, based on what it is trying to achieve with its application logic, how to interpret the received votes, and what action to perform as a response.

This chapter describes the first client application, Clicker++, which is a re-engineered version of the vendor-provided i>clicker software functionality that supports basic classroom voting. It not only fixes some design shortcomings by introducing a new “View by Group” feature, but also adds more powerful visualization components, such as a Participation Bar and Performance Bars, to help both instructors and students qualitatively evaluate in-class performance.

4.1 Basic Clicker++ Functionality

Clicker++ is intended to be a complete replacement for the i>clicker software. It has been programmed in Java, a language that works on many different platforms. The i>clicker vendor provides both an in-classroom application, which is what Clicker++ replaces, and i>grader, an application that takes the output of the i>clicker software and generates documents assessing each student’s performance. The interface between the classroom application and i>grader is through standard
4.2 Improved Histograms

The histograms in the original software show the current number of votes for each choice. During class, we found that although students’ choices may vary a lot when voting starts, as time goes by students who voted the choices with a minority of votes will almost always change their minds and vote for the most popular choice, even though sometimes that choice is incorrect. This phenomenon might be explained by the fact that in-class participation performance will often become part of a student’s final mark for the course (if an instructor has elected to do this), and students may think that it is always safer to go with the majority. This strategy holds little obvious benefit for the learning process, and it discourages the students from thinking independently.

We realized that this happens because the application always shows the number of supporters for each choice, thus revealing the most popular choice. Instructors can of course turn off the histogram, but then students receive no feedback on the voting process. The problem this raises is that there are currently only two extremes for feedback: none at all or complete information about how students are collectively voting.

To solve this problem, a new type of histogram visualization was added, which shows votes by groups of students, instead of by choice. As shown in Figure 4.1, each vertical bar represents a lab group. The height of each bar indicates the percentage of students who have voted. The solid area within the bar indicates the percentage of students in the group whose choices were correct, and the light area within the bar indicates the percentage of students whose choices were incorrect.
4.3. Specifying Groups

In this way, students’ performance is visually available without the identify of the most popular choice being given away.

By default, the entire class is assumed to be in the same group. But the software supports multiple groups, such as for lab sections or other divisions of the full class.

The extra visual hints provided by the enhanced histograms can be beneficial for both instructors and students to quantitatively assess students’ performance. The group-based histogram can be also used to encourage a mild form of competition between lab sections, for example, by letting students see how well their section is doing compared to other sections, both in terms of responses (how many have voted) and accuracy (how many have the correct answer).

The group-based nature of the histograms can be used to “fine tune” the degree of competition because individual students can benefit from how well their groupmates do, not just from their own performance. For example, it is possible for the instructor to set the following rules: all students in a lab group will get participation marks only if 75 percent of the students in the group contribute votes, but in order to get a performance mark, 80 percent of those who have voted must have selected the correct answer.

To make these types of rules visually clear during voting, Participation Bar and Performance Bars were added to the histograms (see Figure 4.1). All groups share the same Participation Bar, which is set to a threshold of 75 percent, as can be seen at the rightmost part of the figure. Only groups that reach the Participation Bar will get participation marks. Additionally, each group has its own Performance Bar. Their values are set to 80 percent (not shown in the figure). All students in a group will get performance marks only if the solid part of the group’s histogram exceeds the Performance Bar.

4.3 Specifying Groups

To support the “Viewing by group” feature, a registration file maintaining the student-group information is required. It can be loaded when the application starts running. Some students own more than one remote and they use them interchangeably. If votes are recognized solely by their remote ID, students who vote with two different remotes will have more say. Thus, we need to keep the student-remote mapping as well (which the vendor-provided software also does).

To support all these, a registration file is created for each course, which contains a list of tuples: student ID (or equivalent unique identifier if student ID is confidential or unavailable), remote ID, and lab group. This file allows the system to make use of group information, and ensures that votes are associated with students instead of remotes. It is the instructor’s responsibility to provide this file.
Should the registration file not be available, the application will tell different votes apart solely by looking at the remote ID. This backwards compatibility is useful when dealing with cases like seminar talks or conference workshops, where remotes are distributed to the audience, so it is not likely that someone has two of them. Requiring registration for these short-term, ad-hoc voting sessions is rarely worth the effort.

As with the vendor-provided software, instructors must specify the correct answers for each vote if they want to provide real-time feedback about performance. Answers are not required for participation feedback.

Figure 4.1: Histogram by lab group. Bar height represents the percentage of students who have voted; solid part represents the percentage of those who contributed correct answer, and the light part represents those whose answers were incorrect. Participation Bar and per-group Performance Bars are shown as well.

### 4.4 Classroom Beta Testing

Clicker++ was used in the UBC course CPSC 260, Data Structures and Algorithms for Computer Engineers, during the first term of the 2012-2013 school year. The application handled about two hundred remotes without any problems. No formal evaluation was conducted, but no problems were detected and it is likely that students did not realize that new software was being used (except when the new visualizations were shown).

While we were able to handle the volume of clicks that were coming in, we met with an unanticipated problem while using the system in the lecture hall: light colors in the interface (e.g., yellow, pink, etc.) that rendered nicely on PC screens were washed out when displayed using the projectors in the lecture hall. Researchers who develop applications that rely on projectors and large screens should thoroughly test all of the visualizations on their intended displays to ensure all the
4.4. Classroom Beta Testing

colors look correct and are distinguishable. Support for alternate color palettes might also be useful.
Chapter 5

Clic^in: Motivation, Architecture and Design

Projected presentation slide is a popular way of displaying instructional material to the class. However, it lacks interactivity. Student response system (SRS) is a big step over traditional classroom teaching and learning approach in the sense that it provides a way for students to participate in the learning process beyond simply listening to the instructor and asking questions when they are confused. However, what can be done with the current i>clacker system is still very limited, mainly completing multiple-choice questions. Additionally, an SRS and a presentation application run in two separate processes independently, which fails to provide seamless user experience. This chapter presents Clic^in, a gamified education application that provides more interactivity, combines both presentation slides and interactive modules (called gamelets) and makes better use of extra screen real estate.

5.1 Motivation

The goal of education gamification is to introduce game elements into regular class to encourage students to try out new concepts and ideas right in the class, to help them gain hands-on problem solving skills, to facilitate collaboration, and to provide real-time feedback about their performance. These are the core ideas in mind when developing Clic^in. We hope that introducing gamelet will increase the sense of in-class participation among students, and will help them to understand the course material in a faster and better way.

Another motivation behind Clic^in is to better exploit extra screen real estate. Nowadays, more and more classrooms are equipped with two or even more projectors and screens. However, in most cases instructors just use one screen, or use more by simply duplicating the current slide across all screens. This duplication provides no extra benefit to both the instructor and the students. Contrarily, displaying different material on different screens may greatly increase the content throughput, thus reducing unnecessary context switch. Previous work on Multi-
Presenter [29] showed how displaying different slides on different screens can be beneficial.

Finally, switching between static presentation slides and gamelets in a class might be both time-consuming and interruptive. Clic^in was designed so that it contains a sequence of activities, each of which can be either a static slide or an interactive gamelet. Combining slides and gamelets together in one application helps the instructor to present and illustrate course material in a natural and organized way.

Clic^in was first designed and developed in [33]. Modifications were later applied to better support the in-class activities of UBC CPSC 260, Data Structures and Algorithms for Computer Engineering. This is a second-year computer science course targets students from the Faculty of Applied Science. Two main topics are basic data structures, such as linked list, binary search tree (BST), and associated operations, such as inserting a node into a linked list or a BST. Students are required to understand how to write correct and fast code for these operations. While letting the instructor to explain the mechanism of these operations is essential, we would also like to ask students, though not to write code directly in the class, but to complete gamified exercises that simulates above operations. The i>clicker system is a perfect platform because it can get real-time input from a large number of students at the same time. With the help of a large shared display, class-wide participation, and group competition are also made available.

5.2 Architecture Overview

Figure 5.1 shows the architecture of Clic^in. The major component of Clic^in is presentation, which is designed for a specific number of screens, and is responsible for making wise use of available screen real estate. It detects the number of screens, and figures out how to position and re-size itself to fit into the screens perfectly. In Figure 5.1, two screens are allocated. Presentation is loaded when Clic^in starts running.

Presentation contains an ordered list of scenarios. Each type of scenario was designed to organize content in a different way. For example, one scenario can be used to show only one gamelet and its associated instructions (left-up scenario in Figure 5.1); one scenario is designed for group competition, where the whole screen is equally divided into eight parts, each of which shows a gamelet controlled by a group of students (right-up scenario in Figure 5.1); one scenario shows one single gamelet and its associated histogram showing students’ choice distribution side by side (left-bottom scenario in Figure 5.1); and another scenario can be used to show two slides (right-bottom scenario in Figure 5.1). Currently a slide can
5.3. Design Specification

only be designed as a static image, which is loaded when Clic\textsuperscript{in} starts running. Gamelets are interactive modules. Each type of gamelet was designed to support one type of operation associated with a certain data structure, for example inserting a new node into an existing BST.

The instructor can advance and back scenarios to select the one he/she wants to show. When a scenario is selected, all students’ votes are directed to it. As stated before, a scenario can contain slides or gamelets or both. If there are gamelets contained in the current scenario, they are called active gamelets. The rest scenarios are hidden and in pause state.

5.3 Design Specification

Clic\textsuperscript{in} is written in Java, because of its cross-platform property. Processing [13], a Java based library focusing on simplified graphics drawing, is used to render all the visual components. Another advantage of using Processing is that it allows instructors to quickly sketch and test a gamelet in a light-weight development environment, and later to integrate it into Clic\textsuperscript{in} with very little modification.

An external registration file is used to provide users’ information, including one’s remote ID, user’s role (Instructor, Student or Demonstrator) and lab group if the user is of role Student. Every time the application starts, it reads the registration file and loads all users’ information.

Users with role of Instructor can control the application but cannot interact with the gamelets. The key-action mappings on Instructor’s remote were designed in a way that is consistent with the ones used in the i>clicker software, thus exploiting positive transfer: ‘A’ is a toggle switching between Start and Stop (students can only interact with an active gamelet when the application is in Start mode); ‘B’ is of no use currently; ‘C’ goes to the next scenario; ‘D’ goes back to the previous scenario; ‘E’ resets the current scenario (so that students can practice the same gamelet one more time). For remotes owned by users’ of role Student or Demonstrator, each key’s action is different depending on what the active gamelets is.

When a vote reaches the Clic\textsuperscript{in} application, presentation first determines the role of its contributor by looking at its remote ID and the registration information. If it is an Instructor, the vote is interpreted immediately, based on the vote’s choice. Otherwise it is a user of role Student or Demonstrator who contributes this vote, which is further passed to the current scenario if the application is in Start mode. The current scenario is then responsible for processing all the incoming votes. How the votes are processed, and how different interaction modes can be achieved with Splitter, Filter and Thresholder will be described in the next chapter.
Figure 5.1: Overview of Clic^in architecture.
Chapter 6

Interacting with a Gamlet

As mentioned in the previous chapter, a gamelet is an interactive module that can be contained in a Scenario. Each gamelet is designed to support one type of operation associated with a certain data structure, for example inserting a new node into an existing binary search tree (BST). Using the BST gamelet as an example, we continue the description of a gamelet in this chapter. We first describe how to interact with a BST gamelet. Then we demonstrate how it can be used to support different interaction modes (concept demonstration, class-wide participation, and group competition) by making use of Splitter, Filter, and Thresholder.

6.1 Binary Search Tree Gamelet

Figure 6.1 shows a BST gamelet. The goal of developing this gamelet is to help students to understand how to correctly insert a node into a BST. When a BST gamelet is shown, there are seven existing nodes in the tree (the root and all the nodes at depth 1 and depth 2). Users are required to insert a total of eight nodes into a BST, all at depth 3. The blue node shown at the top-left corner is the target node waiting to be inserted into the tree. The current node is in green, which is the deepest node in the traversal path. The magenta nodes, along with magenta edges, show the traversal path.

A BST gamelet has five actions. Each of the five keys on the clicker remote maps to one action. As shown in Figure 6.1, the ‘A’ key means selecting the root of the tree; the ‘B’ key and the ‘C’ key are for visiting the left and right child of the current node; the ‘D’ key and the ‘E’ key are used to insert the target node as a new left child or a new right child of the current node. The mappings are always explicitly displayed beside the gamelet.

6.2 Interaction Modes

As mentioned in the previous chapter, there are many types of scenarios. Among them three types of scenarios are of most importance, because they are tailored for three types of interaction modes: concept demonstration, class-wide participation,
6.3. Splitter, Filter, and Thresholder

and group competition. These three interaction modes follow a natural way of how an instructor presents the material and how students absorb the knowledge in the class.

A scenario for concept demonstration has its gamelet labeled with “Demo” at the top-left corner, as shown in Figure 6.1. Only a Demonstrator can interact with such a gamelet. A Demonstrator is usually an instructor or a teaching assistant who explains the knowledge (what a BST is in this case) and shows the class how to interact with a gamelet using a remote (how to insert a node into an existing BST based on the mapping provided).

A scenario for class-wide participation has its gamelet labeled with “All” (see Figure 6.2) and only users of a Student role can interact with it. Key-action mapping is displayed beside the gamelet. The whole class contribute to the same gamelet, which makes it easier for the instructor to focus and explain along the entire process.

Finally, a scenario for group competition has multiple gamelets embedded, as shown in Figure 6.3. Each gamelet is labeled with the associated lab section, and controlled only by students belonging to that lab section. During the game, all gamelets run in parallel. Different scenarios were developed to deal with competitions involving different number of groups so that the gamelets are arranged in a visually appealing way.

![Figure 6.1: A gamelet for concept demonstration.](image)

6.3 Splitter, Filter, and Thresholder

In this section we describe Splitter, Filter, and Thresholder in detail and explain how these components can be used to achieve different interaction modes mentioned previously. For a scenario that has more than one gamelet, Splitter duplicates the vote so that each gamelet gets one copy. Filters, each of which is associ-
6.3. Splitter, Filter, and Thresholder

Figure 6.2: A gamelet for class-wide participation.

Figure 6.3: A gamelet for group competition.

ated with one gamelet, is responsible for filtering out votes coming from users who do not have control over this gamelet. Finally, the Thresholder associated with one gamelet maintains a histogram counting how many users select each of the action. As long as the number of supporters of a certain action exceeds a threshold value, that action is triggered and sent to the corresponding gamelet, and the histogram is reset.

A concept demonstration scenario has only one gamelet, so a splitter is not necessary, as can be seen in Figure 6.4. All votes are sent to the Filter directly. Because only a Demonstrator is allowed to interact with a gamelet embedded in a concept demonstration scenario, all the votes contributed by Students are discarded. The Demonstrator’s vote then reaches the Thresholder. Because usually there is only one user, either the instructor or the teaching assistant, controlling the gamelet, the threshold value of the Thresholder is set to one. This means every vote from a Demonstrator will trigger the corresponding action of a gamelet.

A class-wide participation scenario also has one gamelet, which makes a split-
6.4 Examples of Other Gamelets

The BST gamelet was used as an example through this chapter to explain what a gamelet is, and how it can be used to support different interaction modes using Splitters, Filters, and Thresholders. Other gamelets were also developed to include more data structures. In this section we present the linked list gamelet to demonstrate how to simulate the process of linked list insertion.

The linked list gamelet presents a target node and an existing linked list, whose nodes are sorted in alphabetical order, as can be seen in Figure 6.7. Students are required to insert the target node into the existing linked list so that the new linked list still maintains the alphabetical order. In this example, the node “wow” should be inserted between “foo” and “zap”.

The key-action mapping for the linked list gamelet is as follows. The ‘A’ key selects the head of the linked list as the current node. The ‘B’ key is of no use. The ‘C’ key is used to make the current node move one step forward. The ‘D’ key redirects the pointer of the current node to the target node, while the ‘E’ key does the opposite by redirecting the pointer of the target node to the current node.
6.4. Examples of Other Gamelets

Figure 6.4: Lifetime of a vote in a concept demonstration scenario. The threshold value is 1. Choice of each vote is displayed. The grey vote comes from a Demonstrator; yellow, blue and green votes come from Students who belong to Lab A, Lab B and Lab C respectively.

Figure 6.5: Lifetime of a vote in a class-wide participation scenario. The threshold value is 3.
6.4. Examples of Other Gamelets

Figure 6.6: Lifetime of a vote in a group competition scenario. The threshold value is 2.
6.4. Examples of Other Gamelets

We demonstrate how to complete the linked list insertion as a Demonstrator using a clicker remote in Figure 6.8. The process always starts from the head by pressing the ‘A’ key, as shown in (a), where the head of the linked list is highlighted in red. Then the current node moves forward with three consecutive ‘C’ keys till the current node is larger than the target node alphabetically (“zap” in this example), as shown in (b), (c), and (d). Then the Demonstrator would press the ‘E’ key so that “wow” points to “zap”, as can be seen in (e). However the node “foo” does not point to “wow” yet. So the demonstrator goes back to the head again (shown in (f)), traverses the linked list until it reaches “foo” (shown in (g) and (h)), and presses a final ‘D’ key so that “foo” points to “wow”.

We now have explored all the details of gamelets from a software engineering point of view. However, we are not sure if interacting with a clicker remote may cause additional cognitive load, and if yes, how much is the cognitive load compared to other devices. In the next chapter we describe an experiment measuring the cognitive load of interacting with a gamelet. Although we are referring to gamelet in general, the BST gamelet is used in the experiment. The experiment helps us better understand and evaluate the design of a gamelet from a HCI perspective.
6.4. Examples of Other Gamelets

Figure 6.8: Procedure of linked list insertion
Chapter 7

Cognitive Load in Gamelets

In Chapter 5, we presented a presentation framework Clic-in that supports both static slides and interactive gamelets. In Chapter 6, we described how gamelets are designed and used to increase classroom interactivity by engaging students in game-like activities and we showed the example of a gamelet that illustrates insertion into a binary search tree (BST). A critical aspect of the design of any gamelet is the mapping from clicker keys to actions in the gamelet. As with any interactive system, users (students in the classroom) will need to know the mapping in order to use the system. This could increase the cognitive load on users if the mapping is not carefully chosen.

Traditionally, clickers are mainly used to answer multiple-choice questions. While a multiple-choice question and a gamelet are similar in the way that users perform interactions, namely choosing the key that represents either the correct answer (when completing a multiple choice question) or the correct operation (when interacting with a gamelet), the reasoning processes are fundamentally different. There is only one format for multiple-choice questions, namely one question with multiple answers, no matter what the question is about, and the mapping from answers to keys is always the same because each answer is identified by a key, as shown in Figure 7.1. Students are already familiar with this style of answering because they have taken multiple-choice exams that use a very similar format. However, this is not the case for gamelets. Gamelets can be very different from each other, both in terms of logic and representation. Users have to first understand the state of a gamelet, and then determine what action to take depending on the state. Moreover, the actions in a gamelet do not map to the keys on the remote in a semantically meaningful way. The mapping is often quite arbitrary and thus must be learned for each gamelet.

We assumed that the need to remember key mappings might lead to extra cognitive load because of the need to learn, or re-learn, the key mappings for each gamelet. This chapter describes an experiment that investigated the cognitive load for two different ways of representing the key mapping for the BST gamelet. A follow-up study is also presented to help us understand how participants’ performance changed through practice using both representations.
7.1 Background

The cognitive load of determining the mapping between keys and their actions when playing a gamelet using a remote arises from the need to associate a desired outcome with a specific action and then determine the key to press to achieve that action. This is well explained by Norman’s notions of the Gulf of Execution and the Gulf of Evaluation [34]. We will examine the level of cognitive load imposed by the key mapping using the BST gamelet that was described in Chapter 6.

In the previous chapter, a simple solution was proposed to reduce the cognitive load. The key mapping is explicitly displayed as part of the instructions associated with a gamelet, as shown in Figure 7.2 (a). During the game, while concentrating on the gamelet most of the time, users can occasionally refer to instructions to figure out which key to press for a certain action, in case they are not clear about the mapping. The cognitive load of this split attention is not clear to us, which is one of the research questions that needs to be answered. Another question is whether the different key correspondences in different gamelets also adds cognitive load by requiring users to remember a new mapping for each gamelet. To address the second question, we considered using color-key mapping, which is consistent across all gamelets.

Figure 7.1: A slide showing a multiple-choice question.
Because it is difficult to measure cognitive load directly, we compared performance using a remote against that using a mouse, which does not require key mapping and has no split attention. We also compared a remote against verbal commands, which does not require key mapping but has split attention. Using a remote with the traditional interface requires key mapping and it also results in split attention. Using a remote with the color-coded interface does not result in split attention, yet it requires mapping from the color to the key, instead of showing the key explicitly on the screen. While there are many indicators of cognitive load, we picked task completion time and error rate in this experiment.

### 7.1.1 Design of Interface

**Traditional Interface**

The traditional interface is exactly the same with the one we presented in the previous chapter, as shown in Figure 7.2 (a).

**Color-coded Interface**

The advantage of making use of color to design the mapping is two-fold. First, introducing color into a gamelet does not change its geometric layout. Second, color is used to indicate different keys in Clicker++: red for the ‘A’ key, green for the ‘B’ key, blue for the ‘C’ key, yellow for the ‘D’ key, and purple for the ‘E’ key. Once appropriately established, the color-key mapping is stable in the long run. Although color might be more effective when the keys on a remote are also color-coded (by either painting the keys with different colors or by putting colored stickers), we remained to use the remote shown in Figure 1.1 in this experiment. The color-key mapping used for the color-coded interface is the same with the one used in Clicker++.

The result of applying colors to a BST gamelet is shown in Figure 7.2 (b). All edges connecting a left child are in green. Because green corresponds to the ‘B’ key, it tells a user to click the ‘B’ key in order to go to the left child. All edges connecting a right child are in blue. When a user reaches a node that has less than two children, edges connecting non-existing children are displayed, yellow for left child and purple for right child, indicating that a user needs to press the ‘D’ key or the ‘E’ key for inserting the target node as a new left child or a new right child. Finally, the red edge connecting the target node and the root tells the user to press the ‘A’ key to select the root.

For some gamelets, re-design its interface by introducing color might be problematic, because color channel has been used to convey other information. As mentioned in the previous chapter, in a traditional BST gamelet (see Figure 7.2
7.1. Background

(a)), color indicates the role of a node. A blue node is the target node. A green node is the current node. The purple ones are those that have been visited along the traversal path. Now that the color channel is used to convey the information of which key to press, we need to find a way to represent information that was previously denoted by color, especially which node is the current node. We solved this problem by enlarging the size of the current node so that it becomes visually dominant, as the node M in Figure 7.2 (b). Finally, all nodes that are impossible to reach from the current node become grey, which helps users to focus on the part of the tree they are working on.

Figure 7.2: Graphics interface of a BST gamelet: (a). traditional interface; (b). color-coded interface; (c). interface when using a mouse.

Mouse Interface

The mouse interface used in a BST gamelet was modified based on the traditional interface. It supports interaction using a mouse by clicking inside the node (see Figure 7.2 (c)). To be more specific, clicking at the root means going to the root. Clicking at the left child of the current node means going to the left child, and clicking at the right child means going to the right child. To insert a node, because the spot was empty before insertion, an empty rectangle was displayed to indicate
the area to click at (see the bottom rows of the BST in Figure 7.2 (c)). Clicking at nodes that could not be reached from the current node (e.g., nodes representing the grandchild of the current node) would not have any effect. Clicking at areas outside of the BST had no effect neither.

Verbal Commands Interface

When interacting with a BST gamelet using verbal commands, the graphic interface is exactly the same with the traditional interface, as shown in 6.1 (a). However, the key-action mapping is replaced by command-action mapping. A user tells an assistant what the next step is using verbal commands. There are five commands, each maps to one action: “Root” (go to the root of the tree), “Left” (go to the left child of the current node), “Right” (go to the right child of the current node), “Insert Left” (insert target node as a new left child of the current node), and “Insert Right” (insert target node as a new right child of the current node). Upon hearing user’s command, the assistant will press the corresponding key on the remote as fast as possible: A for “Root”, B for “Left”, C for “Right”, D for “Insert Left”, and E for “Insert Right”.

7.2 Participants

In order to replicate a classroom scenario in which students are often not already familiar with the material being presented, we only recruited participants who had never heard of a BST. A total of 26 participants were recruited. Data from two participants’ results were not used because their raw data files were accidentally deleted during a file transfer process. The remaining 24 participants were all UBC students, 10 males and 14 females. There were 4 master’s students, 3 doctoral students and 1 postdoctoral fellow. All the others were undergraduate students (4 had just graduated with a bachelor’s degree). Participants came from diverse academic backgrounds: 4 from the Faculty of Applied Science, 9 from the Faculty of Arts, 5 from Sauder School of Business, 4 from the Faculty of Science, 1 from the Faculty of Land and Food Systems, and 1 from the Faculty of Forestry. Fourteen participants had used an i>clicker before.

All participants received compensation of $10 for their participation, regardless of their performance.
7.3 Apparatus

The experiment was carried out at UBC in ICICS X521. Each participant sat at a desk, facing a large screen 6 meters away. Both mouse and clicker remote were provided. The screen was about 2.85 meters wide and 2.15 meters high, with resolution of 1280 x 960 pixels. The software ran on a laptop (Gateway T6308C) with Windows 7 Ultimate installed. The Clic^in software, which was described in Chapter 5, was used in the experiment, modified to accept and process mouse input in addition to clicker input, and to keep a record of all the remote key clicks (timestamp and the key clicked) and mouse clicks (timestamp and coordinates). The BST gamelet, which was described in Chapter 6, was modified to also generate the color-coded interface.

7.4 Task and Procedure

After signing a consent form, participants were instructed to carry out four blocks of experiments, followed by a short survey after all blocks were completed. Each block consisted of 5 BSTs, each with 7 nodes and having 8 additional nodes waiting to be inserted into the BST. A trial was the insertion of one node into a BST, so each BST was used for eight consecutive trials before a new BST was encountered.

Participants used four different interaction techniques to complete different blocks. Thus, the overall design of the experiment was: 24 participants x 4 interaction techniques x 5 trees x 8 nodes, a total of 3840 trials. The order of the four blocks was counter-balanced. Before each block started, there was a short training session for participants to practice, in which they were asked to insert two nodes into a BST. The goal of the training session was to help participants better understand how the task should be performed. We avoided longer training session because it might result in significant learning effects. For all tasks, participants were told that accuracy had higher priority over speed.

For error handling, two scenarios were explained to all participants. First, whenever realizing that they traversed in the wrong direction, e.g. they should go left but instead they go right, participants should always correct the error. Second, if a node was inserted at an incorrect place, a red cross would appear over that node. Participants must correct the error first before inserting the next node. In both cases, participants corrected the error by going back to the root and restarting the traversal process. As long as either type of mistake occurred in a trial, that trial was a failed trial. Otherwise it was a successful trial. Whether a trial is successful or not was recorded. Details of each task are described below.
7.4. Task and Procedure

7.4.1 Remote with Traditional Interface
This task measured the cognitive load of interacting with a traditional BST gamelet using a remote. Participants performed this task by using an i>clicker remote. Associated instructions were displayed beside the gamelet.

7.4.2 Remote with Color-coded Interface
This task measured the cognitive load of interacting with the color-coded BST gamelet using a remote. Participants learned the color-key mapping before the experiment. To help them better remember the mapping, a separate training session was carried out before this task, in which a colored rectangle was displayed on the big screen. Participants were told to click the corresponding key based on the color of the rectangle. The training session ended when the error rate of the last 25 trials dropped below ten percent.

7.4.3 Mouse
The goal of this task was to determine the cognitive load of interacting with a BST gamelet when using a mouse, one of the most common input devices. The mouse interface was used in this task.

7.4.4 Verbal Commands
This task measured the cognitive load of a BST gamelet itself without using any device. The verbal commands interface was used, and the experimenter acted as an assistant for the user. We were aware that the completion time measured in this task may not be accurate because it included the time it took for participants to speak out the command and the time it took for the experimenter to respond. However, we assumed that the error rate should be an accurate estimate of the cognitive load when interacting with a BST gamelet without using any physical device.

7.4.5 Post-experiment Survey
In the survey, participants were asked about their background information including gender, major, and which year they were in. They were also asked about their experience using i>clicker, including whether they had used it before, number of courses they took in Winter Term 2012, and how many of these courses required an i>clicker. Participants were asked to provide their general opinion towards gamelet, including whether they have seen/heard of/experienced other ways of using i>clicker in the class besides completing multiple-choice questions, their
7.5. Hypotheses

The main goal of this experiment was to compare the cognitive load of interacting with a BST gamelet using different interaction techniques. More specifically, we were interested in comparing the cognitive load when using a remote and that when using a mouse. Also, we would like to compare the cognitive load of different graphical interface (traditional and color-coded) when using a remote.

We had two primary hypotheses. The first hypothesis dealt with error rate. The error rate was computed as follows. For each interaction technique, each participant performed 8 x 5 insertions. The error rate was obtained by dividing the number of unsuccessful insertions by the number of total insertions, which is 40.

We hypothesized that there would be a significant effect of interaction technique on error rate. More specifically, we expected that using a remote would produce significantly more errors than using a mouse or verbal commands. This is because it is possible for a participant to press the wrong key, although he/she does understand what action to take. For example, a participant wants to go to the left child, and he/she understands that the ‘B’ key should be pressed in this case, but instead the adjacent the ‘C’ key is pressed. Additionally, we expected that when using a remote, the color-coded interface would be less error-prone than the traditional interface.

The second hypothesis was about completion time. The task completion time was computed only for successful trials. The task completion time for a successful trial starts from the moment when the target node appeared on the top left of the screen, to the moment when the node was successfully inserted into the tree.

We hypothesized that there would be a significant effect of interaction technique on task completion time. More specifically, we expected that using a remote would be significantly slower than using a mouse. In addition, we expected that interacting with the color-coded interface when using a remote would be significantly faster than interacting with the traditional interface.

We had one additional hypothesis solely regarding to the BST gamelet, which would help us better understand it. We broke down a successful trial into four steps: selecting root, first branch, second branch, and inserting node. In each step, the current node gets one level deeper. We hypothesized that for all interaction techniques, there would be a significant effect of step on both interaction time and error rate. To be more specific, as the current node gets deeper, it should be faster for participants to make a decision, and the decision should be less error-prone.
7.6 Results and Discussion

This is because when the current node gets deeper, participants were comparing characters closer to each other in the alphabet, which should be simpler (i.e. we assumed that comparing I and J should be simpler than comparing I and F).

To conclude, the following hypotheses were tested in the experiment.

H1: interaction technique does affect error rate. More specifically, using a remote is more error-prone than using a mouse or verbal commands. When using a remote, the color-coded interface is less error-prone than the traditional interface.

H2: interaction technique affects task completion time. More specifically, using a remote is slower than using a mouse. When using a remote, interacting with the color-coded interface is faster than interacting with the traditional interface.

H3: For all interaction techniques, depth of the current node affects error rate. More specifically, when the current node gets deeper, the decision is less error-prone.

H4: For all interaction techniques, depth of the current node affects interaction time. More specifically, when the current node gets deeper, it takes less time to perform a correct action.

7.6 Results and Discussion

We describe the results obtained from the experiment. We first look at the error rate and per-step error rate. Then we present task completion time and per-step interaction time. After that we describe how the participants’ performance improved over time as they gained more experience interacting with a BST gamelet. Finally, feedback from post-experiment survey was presented.

7.6.1 Error Rate

The first dependent variable to look at was error rate. The average error rate for each interaction technique was as follows: 2.0% for mouse, 7.2% for verbal commands, 15.4% for remote with the color-coded interface and 19.4% for remote with the traditional interface. A repeated measure ANOVA revealed that there was a significant effect of interaction technique on error rate ($F_{2.39, 54.9} = 25.1, p < .005, \eta^2_p = .52$). Post-hoc analysis using Tukey’s method showed that when using a remote, regardless of which type of interface was used, the error rate was significantly higher than using verbal commands ($q_{4, 68} = .12, p < .001$ for the traditional interface and $q_{4, 68} = .08, p < .005$ for the color-coded interface). Similar results were obtained between a remote and a mouse. There was a significant difference between a remote and a mouse ($q_{4, 68} = .16, p < .001$ for the traditional interface and $q_{4, 68} = .12, p < .001$ for the color-coded interface). When using a
remote, the two interfaces were not significantly different in terms of error rate ($q_{4,68} = .04, p > .05$). However participants made significantly less errors when using a mouse than when using verbal commands ($q_{4,68} = .04, p < .005$).

**Discussion**

Most results obtained were anticipated, except that using a mouse led to significantly less errors than verbal commands, while there should not be any difference if we assumed that using verbal commands would introduce few errors. The only possibility that gave rise to a 5.2 percent difference could have been that participants sometimes made mistakes when speaking out the command, e.g. intending to go left but saying the command “Right”.

**Conclusion**

To summarize, $H1$ was partially confirmed: interaction technique affects error rate, and using a remote is more error-prone than using a mouse or verbal commands. However, the two types of interface do not significantly differ from each other in terms of error rate.

**7.6.2 Error Rate in Each Step**

We examined the error rate in each step. A two-way repeated measure ANOVA revealed significant effects of both interaction technique ($F_{2,39,54,9} = 25.1, p < .001, \eta^2_p = .52$) and step ($F_{2,25,51.8} = 24.6, p < .001, \eta^2_p = .52$) on error rate. The interaction was significant as well ($F_{4,47,103} = 5.4, p < .001, \eta^2_p = .19$). Figure 7.3 shows the average error rate for each interaction approach in each step.

**Discussion**

From the above results, we concluded that depth of the current node does affect error rate. In Figure 7.3 we found that the first step, which was selecting root, had the lowest error rate for all interaction approaches. This could be explained by the fact that selecting root was the easiest step, which did not involve much decision-making. For all interaction techniques except mouse, the error rates at the second branch were lower than the ones at the first branch, which was consistent with our hypothesis. It was interesting to see how the error rates in the last step changed differently among different interaction techniques. For mouse and verbal commands, the error rates in the third step and the last step were quite close ($q_{4,68} = .13, p > .05$ for mouse and $q_{4,68} = .083, p > .05$ for verbal commands). However, when remotes were used, significant increase was observed ($q_{4,68} = 1.12, p < .01$).
7.6. Results and Discussion

Figure 7.3: Error rate for each interaction technique and step combination.

for color coded interface and $q_{4.68} = 1.08, p < .01$ for traditional interface). That is to say, participants tended to make more mistakes in the last step when using a remote.

To understand why the last step was more error-prone when using a remote, results in each step were further broken down and were categorized into bins according to which key should press and which key was actually pressed. Table 7.1 (a) to 7.1 (d) show the broken-down details for both the traditional interface and the color-coded interface when using a remote.

From the table we can see that there are two main reasons why the last steps had higher error rates for both interfaces. First, in the third step (second branch), few participants pressed the ‘D’ key and the E ‘key’. This is understandable because the ‘D’ key and the ‘E’ key are only used when performing the last step (inserting node). On the contrary, in the last step more participants pressed the ‘B’ key and the ‘C’ key, probably because they forgot to switch to the ‘D’ key and the ‘E’ key when trying to insert a new node.

Second, we found that the error rate caused by the confusion between the ‘D’ key and the ‘E’ key in the fourth step (should insert as a new left child but actually inserted as a new right child, and vice versa) was 3.981% for the traditional interface and 2.217% for the color-coded interface, both of which were higher than the error rate caused by the confusion between the ‘B’ key and the ‘C’ key in the third step (should branch left but actually branched right, and vice versa), which was 1.981% for the traditional interface and 1.6% for the color-coded interface.

The second reason mentioned above could be treated as a side-effect of switching from the ‘B’ key and the ‘C’ key to the ‘D’ key and the ‘E’ key. This can be proven by the fact that the error rate in the last step did not increase for interaction technique that does not involve any device (e.g. verbal commands) or interaction
technique requiring a device but does not need such a switch (e.g. mouse).

Conclusion

To summarize, H3 was rejected: depth of the current node affects error rate. However the error rate does not necessarily decrease as the current node gets deeper. It depends not only on which step the current node is in, but what the interaction technique is.

7.6.3 Completion Time

Completion time was measured in millisecond. As expected, a repeated measure ANOVA showed that there was a significant effect of interaction approach on completion time ($F_{3,69} = 5.74, p < .005, \eta^2_p = .20$), with mouse being the fastest (6446), followed by verbal commands (7517), then remote with the color-coded interface (8230), and finally remote with the traditional interface (8751). Post-hoc analysis using Tukey’s method revealed that a mouse was significantly faster than a remote ($q_{4,68} = 1783, p < .005$ for the color-coded interface, and $q_{4,68} = 2304, p < .005$ for the traditional interface). Although interacting with the color-coded interface was faster than with the traditional interface, the difference was not significant ($q_{4,68} = 521, p > .05$).

Conclusion

From the above results we conclude that H2 was partially confirmed: interaction technique affects completion time. Using a remote is slower than using a mouse for interacting with a BST gamelet. However, the advantage of the color-coded interface over the tradition interface is not significant.

7.6.4 Interaction Time in Each Step

We the looked at the interaction time in each step. A two-way repeated measure ANOVA revealed a significant effect of interaction technique ($F_{2.59,59.6} = 5.74, p < .005, \eta^2_p = .20$) and step ($F_{2.33,53.5} = 18.5, p < .001, \eta^2_p = .45$). There was a significant interaction effect as well ($F_{4.85,112} = 29.3, p < .001, \eta^2_p = .56$). Figure 7.4 shows the average interaction time for each interaction technique in each step.

Discussion

We first share some discoveries obtained from the graph. We found that participants spent longest time in the first step only when using a mouse. This was
7.6. Results and Discussion

<table>
<thead>
<tr>
<th>Should Press</th>
<th>Pressed A</th>
<th>Pressed B</th>
<th>Pressed C</th>
<th>Pressed D</th>
<th>Pressed E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>935 954</td>
<td>12 0</td>
<td>12 3</td>
<td>0 2</td>
<td>1 1</td>
</tr>
<tr>
<td>B</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>C</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>D</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>E</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
</tbody>
</table>

(a). Selecting root.

<table>
<thead>
<tr>
<th>Should Press</th>
<th>Pressed A</th>
<th>Pressed B</th>
<th>Pressed C</th>
<th>Pressed D</th>
<th>Pressed E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>B</td>
<td>5 13</td>
<td>444 449</td>
<td>20 16</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>C</td>
<td>5 16</td>
<td>40 34</td>
<td>414 426</td>
<td>7 0</td>
<td>0 0</td>
</tr>
<tr>
<td>D</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>E</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
</tbody>
</table>

(b). First branch.

<table>
<thead>
<tr>
<th>Should Press</th>
<th>Pressed A</th>
<th>Pressed B</th>
<th>Pressed C</th>
<th>Pressed D</th>
<th>Pressed E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>B</td>
<td>4 0</td>
<td>405 421</td>
<td>13 9</td>
<td>4 0</td>
<td>0 0</td>
</tr>
<tr>
<td>C</td>
<td>0 0</td>
<td>4 5</td>
<td>424 436</td>
<td>4 1</td>
<td>0 3</td>
</tr>
<tr>
<td>D</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>E</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
</tbody>
</table>

(c). Second branch.

<table>
<thead>
<tr>
<th>Should Press</th>
<th>Pressed A</th>
<th>Pressed B</th>
<th>Pressed C</th>
<th>Pressed D</th>
<th>Pressed E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>B</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>C</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
<td>0 0</td>
</tr>
<tr>
<td>D</td>
<td>1 3</td>
<td>9 9</td>
<td>3 7</td>
<td>393 408</td>
<td>7 6</td>
</tr>
<tr>
<td>E</td>
<td>0 0</td>
<td>3 1</td>
<td>6 6</td>
<td>26 13</td>
<td>381 404</td>
</tr>
</tbody>
</table>

(d). Inserting node.

Table 7.1: Break-down of results in each step for traditional interface (left in each cell) and color-coded interface (right in each cell).
7.6. Results and Discussion

Figure 7.4: Interaction time for each interaction technique and step combination.

because the cursor was located at the bottom of the screen after one trial was over, and it took time for participants to move the cursor to the top and to selected the root when inserting the next target node. Unlike other interaction techniques, when using verbal commands the time spent at different steps was not significantly different ($F_{4,85,112} = 1.07, p > .05, \eta^2_p = .045$). This was because verbal commands was the only interaction technique that did not require participants to use any device. Instead it was the experimenter who applied all commands using a remote and the experimenter was equally fast to respond to all verbal commands. Finally, selecting root took the least time among all steps when using a remote, which was expected because selecting root is straightforward.

We now look at how the interaction time changed as the current node got deeper. We excluded verbal commands in the following analysis because as mentioned above, there was no significant effect of step on interaction time. For the rest three interaction techniques, significant drops of interaction time were observed from the first branch to the second branch ($q_{4,68} = 368, p < .001$ for mouse, $q_{4,68} = 477, p < .005$ for remote with traditional interface, $q_{4,68} = 1068, p < .001$ for remote with color-coded interface). This is consistent with our hypothesis: it takes less time to perform an action when the depth of the current node increases. However if we look at the time it took to perform the last step, contrary results were observed. No significant decrease was found for all interaction techniques ($q_{4,68} = 15.3, p > .05$ for mouse, $q_{4,68} = 99.4, p > .05$ for remote with traditional interface, $q_{4,68} = 49.2, p > .05$ for remote with color-coded interface). Thus, we cannot accept our hypothesis.
Conclusion

To conclude, $H4$ was rejected: depth affects interaction time. However the interaction time does not necessarily decrease as the current node gets deeper. It depends on both where the current node is and what the interaction technique is.

7.6.5 Performance Improvement

We were also interested in how participants’ performance improved as they were more and more experienced in interacting with a BST gamelet. We depict the error rate and interaction time for each BST in Figure 7.5 and Figure 7.6. We found that error rate did not change much over time ($F_{4,92} = .31, p > .05, \eta^2_p = .013$), as can be seen in Figure 7.5. However, there was a significant drop of completion time ($F_{4,92} = 5.48, p < .001, \eta^2_p = .192$). Figure 7.6 shows that it took longest time for participants to complete the first BST for all interaction techniques. Later on they spent less and less time for each BST.

![Error rate from the first tree to the last (fifth) tree when using different interaction techniques.](image)

Figure 7.5: Error rate from the first tree to the last (fifth) tree when using different interaction techniques.
7.6. Results and Discussion

Figure 7.6: Interaction time from the first tree to the last (fifth) tree when using different interaction techniques.

In order to explore the training effects more thoroughly and understand the lowest possible error rate and the shortest interaction time when using a remote, a follow-up between-subjects experiment was carried out. This time newly recruited participants were asked to perform the same task with a remote using either the traditional interface or the color-coded interface. Each participant was required to complete a total of sixteen trees. Ten participants were recruited, and each interface was tested by five participants. Figure 7.7 and Figure 7.8 show the averaged error rate and interaction time for each tree.

Figure 7.7: Error rate from the first tree to the last (sixteenth) tree when using color-coded interface and traditional interface.
7.6. Results and Discussion

Figure 7.8: Interaction time from the first tree to the last (sixteenth) tree when using color-coded interface and traditional interface.

Discussion

Figure 7.7 shows that for both interaction techniques, from the forth BST the error rate started oscillating between 5% and 25%, with an average value of 16% for the traditional interface and 13% for the color-coded interface. On the contrary, one can found in figure 7.8 that it took longer time for the interaction time curves to converge for both techniques. The best performance for both interaction techniques appeared in the last tree, 5595 milliseconds for color-coded interface and 5251 milliseconds for traditional interface. It is possible to get smaller values if we increased the task size, yet a value of 5 seconds should be a good estimate of the best performance for both interaction techniques. To summarize, it took longer time for participants to achieve their highest speed.

7.6.6 Results from Survey

We present survey results provided by our participants. When asked whether there were other ways of using i>clicker in the class besides completing multiple-choice question, one participant mentioned that some SRS remotes had number keys so that students could answer arithmetical questions. Another participant mentioned that instructors used i>clicker to call the roll. When asked if gamelet was a good way to learn BST in class, 7 participants displayed an extremely positive attitude, 10 for mildly positive, 6 for neutral and 1 for mildly negative. Participants who maintained an extremely positive attitude or positive attitude were asked to explain their reasons. The most popular reason, mentioned by 12 participants, was that gamelet helped increasing classroom participation and sense of involvement. A
total of 11 participants mentioned that the interface was properly designed so that it was easy to see and understand the result of interaction on the big screen. Additionally, 8 participants valued ClicIn’s combination of learning and entertainment, and 7 participants demonstrated support because most students already owned an i>clicker remote. One participant who demonstrated a negative attitude explained that it would take too much time in the class to learn how to interact with a gamelet. The time could be better used to cover material that was more helpful to students.

Participants also provided feedback regarding the design of both the interface and the device. Three participants mentioned that it was easier for them to memorize the mapping and to click the keys when holding the remote horizontally. Two participants expected a mobile application that could substitute the physical remote. Two participants argued that it was important to include an “undo” functionality so that one did not always need to go back to the root to restart the entire trial in case he/she made a mistake. One participant mentioned that gamelet was a good idea to support interaction besides completing multiple-choice question. One participant pointed out that for the color-coded interface, he/she preferred a different color schema that was similar to the one in the rainbow, which might be easier to memorize. Although that participant also pointed out that different users might have different ideas in terms of what color schema worked out best. One participant believed that it would be useful to color-code the remote as well (by putting colored stickers) so that users could refer to the color-key mapping more easily. One participant mentioned that the color-key mapping was hard to remember and another one pointed out that explicitly showing the mapping information somewhere on the screen would be more convenient.

### 7.7 Conclusion

In this section we summarize all the results obtained from this experiment. There was a significant performance gap between a remote and a mouse when interacting with a BST gamelet. Using a remote with the traditional interface leads to 17.4% more errors than using a mouse. The percentage drops down to 13.4% for the color-coded interface. For completing one trial, a remote is about two seconds slower than a mouse. We also compared two types of interface when using a remote, i.e. the traditional interface and the color-coded interface. Although better performance was observed for the color-coded interface in terms of both error rate and interaction time, yet the difference was not statistically significant in both cases. For a BST gamelet, the error rate and interaction time in each step varied, depending on both the depth of the current node and the interaction technique. Additionally, as participants gained more training experience, they were able to
7.7. Conclusion

improve their performance, in terms of both error rate and interaction time.

Although it is true that in this experiment a mouse is superior than a remote in terms of both error rate and completion time, we would like to justify the usage of the remote in a real classroom situation. Students bring remotes to the class, but few students will bring a mouse because nowadays most laptops include a touchpad. What’s more, we would like to emphasize that Clic^in was not designed for a single user. Instead the support for collaboration is one of the most important design features we had in mind. However it is technically impossible for a large group of students to control the same cursor using many mice at the same time in a class. Additionally when deploying Clic^in in a real class, we found that most time was spent waiting the number of students choosing the correct answer exceeding the threshold, which made the two seconds delay almost negligible. The same case applies for error rate. Although there were some students who clicked the wrong keys, as long as the majority did it correctly, the game could still proceed without any problem.
Chapter 8

Selection Tool: Slide Navigation and Content Highlighting using a Remote

Presentation slides are a popular way to share information in a classroom. It has several advantages over a traditional blackboard, such as off-line authoring, easy editing, and quick distribution. During a lecture, the presentation software is usually controlled by an instructor, not by students. Students’ lack of direct control may be inconvenient sometimes, especially when a student wants to perform slide navigation or to refer to the content on a certain slide. In this chapter, we first provide a detailed description of the research question. We then present our solution, a tool for slide navigation and content highlighting that lets students perform these actions when enabled by the instructor. We demonstrate how the tool can be used from a student’s perspective. We explore the cognitive load of the content highlighting tool in the next chapter.

8.1 Motivation

In this section we explain the motivation for the Selection Tool with two scenarios. The first scenario shows that sometimes it is useful to allow students to have direct control over slide navigation. The second scenario explains why visually highlighting the content on a big screen can benefit both the instructor and the students.

Imagine the following scenario. An instructor is talking about the concept of a BST, and the presentation software is showing a slide of a BST on the big screen. One student is confused by the difference between a BST and a regular binary tree. However, the concept of a binary tree is not described in the current slide, but in one of the previous slides. So the student would like to go back to that slide and have a detailed look. The student raises his/her hand and tells the instructor to back the slide. Because there is more than one slide that talks about a binary tree, the student does not know exactly which one he/she wants. Thus, the student has to ask the instructor to back one slide every time if necessary. This process
may take a long time, with the whole class sitting in the classroom waiting. This process can be sped up if the student who asks a question can navigate the slide by himself/herself, which is faster than telling the instructor to do it.

The next scenario explains why visually highlighting the content on a big screen can be beneficial. Think about the following scenario. In a machine learning class, the instructor is deriving an equation. At the end, the current slide is full of mathematical terms and symbols (see Figure 8.1). One student realizes that there is a mistake on the slide: one theta misses a subscript. The student raises his/her hand and tells the instructor. However, the instructor, along with the whole class, has no idea which theta the student is referring to, because there are more than twenty theta letters on this slide. So the instructor would ask if the student can be more specific in terms of where that theta is. The student then spends some time determining which line and which term it is in, and tells the instructor that the theta locates in the sixth line, and it is the first theta in the forth term inside the parenthesis. The instructor then follow what the student says and find out the theta. Meanwhile, any student who are taking notes also have to repeat this process to modify their notes. This process can be very time-consuming. One solution can be letting the student to highlight that theta on the big screen. Because the student knows exactly where the theta locates, this process should be more efficient. Additionally, providing visual feedback on the big screen allows everyone to see clearly where the theta is.

Both the slide navigation task and the content highlighting task require an interaction platform through which a student can provide his/her input. Now that the i>clicker system are widely used in the class, we explore approaches that support slide navigation and content highlighting using an clicker remote.

8.2 Overview of the Selection Tool

We built a tool that enabled a student to navigate slides and to highlight content using a remote. Before going into details of how the tool works, we’d like to describe how to grant control only to the student who raises a hand or asks a question. As described in Chapter 2, the base station only has two modes, “accepting” and “not-accepting”. Granting control to one student means putting the base station into “accepting” mode, thus granting control to all the other students at the same time. We assume that this works most of the time because students will follow social convention: they will let their peer who asks a question control the tool. However, this will be problematic when other students accidentally hit the keys on their remotes. To solve this problem, we adopted a “first come, first only” strategy: when the instructor grants control to students (by pressing the ‘E’ key on his/her
8.2. Overview of the Selection Tool

remote, which will put the base station into “accepting” mode), the student who clicks first will be granted control. Votes from other students can still be received by the base station, but they are discarded by the Selection Tool. The instructor reclaims control by pressing the ‘E’ key again, which puts the base station into “not-accepting” mode. One can see that the process of control competition still relies on social convention, but this is much more reliable than allowing everyone to control the tool during the entire navigation or highlighting process.

In summary, an instructor presses the ‘E’ key to allow the students to compete for control. The student who clicks first will be granted control. The student presses ‘C’ key and ‘D’ key to forward and back a slide. ‘E’ key is for initiating the content highlighting tool (hereinafter referred to as the highlighting tool). During any time point of slide navigation or content highlighting, control can be reclaimed by the instructor by pressing the ‘E’ key again. The next section is dedicated to the design of highlighting tool.
8.3 Design of the Highlighting Tool

Various kinds of information can be displayed on a slide, including text, image and video, etc. In this project we focused on text highlighting, because it is easier to refer to images and videos verbally. Text has many forms of representation as well, including plain text (which are usually sentences or paragraphs), tables, program segments, and equations, etc. The smallest semantically meaningful unit is a word, and it also should be the smallest unit to highlight as well. A word is usually a sequence of characters, sometimes mixed with numbers. A word can appear almost anywhere on the slide.

Highlighting a word using a remote, which only has five keys, is very challenging. Keys only support discrete operation, while highlighting a target is usually performed by a mouse, which can move continuously in the real world (although deep down the movement of cursor is still discrete, since pixel is the smallest unit on the screen). We now propose a solution that recursively “Shrink” the highlighted area, followed by fine tuning using “Translate” to complete the highlighting process.

Figure 8.2 shows one complete highlighting process. Once the highlighting tool is initiated, the whole screen is highlighted, as shown in Figure 8.2 (a). The highlighted area contains two Separators, which split the whole area into four parts. Depending on which part contains the target, or which part has the largest portion of the target, that part is selected by pressing the key indicated by the label located on its corner. In this example the word “Target” is in part D, so the ‘D’ key should be pressed. Now only one part of the screen is highlighted (thus the highlighted area “ Shrinks”, as can be seen in Figure8.2 (b)). This process continues for several iterations until the size of the highlighted area is similar to the size of the target, although the highlighted area may be slightly off the target (see Figure8.2 (e)). To make them better overlap, a user presses the ‘E’ key, which brings the tool into “Translate” mode. In “Translate” mode, all Labels change their positions: now first four keys represent the direction of translation (see Figure8.2 (f)). By hitting one of these keys, the highlighted area translates accordingly. Once the highlighted area almost covers the target, the user can press a final ‘E’ key to finish the highlighting process.

We are not yet sure of the cognitive load of our highlighting tool in terms of interaction speed and error rate. Next chapter describes an experiment that investigates the cognitive load involved in this process.
8.3. Design of the Highlighting Tool

Figure 8.2: Content highlighting process: (a) when the process starts, the entire screen is highlighted; (b) ‘D’ key is pressed to highlight the bottom right part; (c) ‘A’ key is pressed to highlight the top left part; (d) ‘D’ key is pressed to highlight bottom right part; (e) ‘C’ key is pressed to highlight bottom left part; (f) ‘E’ key is pressed, which brings the tool from Shrink mode to Translate mode; (g) ‘A’ key is pressed and highlighted area moves up; (h) ‘C’ key is pressed and highlighted area moves right.
Chapter 9

Performance Assessment of the Content Highlighting Tool

In the last chapter we described the Selection Tool, an application that allows a student to navigate slides and to highlight contents. However, we were not sure if the highlighting tool is fast and accurate enough to complete the highlighting task. An experiment was carried out to evaluate the usability of the highlighting tool. The experiment compared our highlighting paradigm with a mouse, in terms of both interaction speed and error rate.

We were aware that relying on a remote, our highlighting tool was slower than a mouse. However, because mouse is the most popular and efficient input device for the single user highlighting task, such a comparison may help us better understand the discrepancy, and to help us discover any advantage of our highlighting tool over a mouse. Additionally, we would like to examine the error rate when using the highlighting tool, as well as how the performance changes when highlighting targets with different sizes. Finally, because our highlighting paradigm is very different from that using a mouse, we would like to examine some of the aspects that is unique to the highlighting tool, such as the effect of number of hands used on performance, etc. This chapter describes the experiment and reports the results.

9.1 Participants

A total of 25 participants were recruited for the study. Among them, 4 participants’ results were not used, because the system parameters for their experiments were incorrectly set. One participant misunderstood the instruction and his/her result was also discarded. The rest 20 participants were all UBC students, 15 males and 5 females. There were 11 master’s students and 4 doctoral students, with the rest five being undergraduate students. 11 of them came from the Faculty of Science, 6 from the Faculty of Applied Science and 3 from the Faculty of Arts. Nine participants had used an i>clicker remote before.

All participants were compensated with $10 for their participation, regardless of their performances.
9.2 Apparatus

The experiment was carried out at UBC in ICICS X521. Each participant sat at a desk, facing a large screen 6 meters away. Both mouse and clicker remote were provided. The screen was about 2.85 meters wide and 2.15 meters high, with resolution of 1280 X 960 pixels. The software ran on a laptop (Gateway T6308C) with Window 7 Ultimate installed. A customized version of the highlighting tool described in the previous chapter was used. It was modified so that key clicks were recorded, including which key was pressed and the timestamp. The software also kept record of information such as target size, target location, and when and where a mouse click took place.

9.3 Task and Procedure

After signing a consent form, participants were told to complete two blocks of experiments, followed by a short survey after both blocks were completed. In both blocks participants were asked to highlight targets on the screen, but with different devices, either a mouse or a clicker remote. The order of the two blocks was counter-balanced. After instructions were explained, participants were asked to practice until they fully understood how to use both devices.

Although in real case the task would be highlighting a word on a slide, the scenario in this experiment was simplified to highlighting a rectangle on a blank slide. At any time point there was only one rectangle on the slide, which appeared randomly at any location. There were three sizes of rectangles, small, medium, and large.

To highlight a target using a mouse, a participant moved the mouse and clicked inside the rectangle. We believe this is appropriate because the most common approach to highlight a word on a slide is to move the mouse so that the cursor hovers over the target. Once the participant clicked, this target would disappear, regardless of whether the click was inside or outside of the rectangle, and a new target would appear at another random position on the slide. All participants were told that accuracy had higher priority over speed.

To highlight a target using the highlighting tool, participants were told to shrink the highlighted area until they felt its area was close to that of the target. Then participants switched to Translate mode and moved the highlighted area until it roughly covered the target. The criteria was still the same, accuracy came first, followed by speed.

Similar to the survey of the previous experiment, at the end of this experiment participants were asked about their background information, their experience with
9.4 Hypotheses

We had two hypotheses regarding the comparison between a mouse and the highlighting tool. First, we hypothesized that there would not be a significant effect of device type on error rate. Namely, we assumed that the highlighting tool could perform as well as a mouse in terms of error rate since participants were told that accuracy had higher priority over speed.

Error rate was computed by dividing the total number of unsuccessful trials by the total number of trials. An unsuccessful trial is defined differently for different devices. For a mouse, a trial is unsuccessful if the point clicked is outside of the rectangle. For the highlighting tool, the process is unsuccessful if it meets any of the following three conditions: 1. at any time of Shrink, the highlighted area fails to cover any part of the target (see Figure 9.1 (a)); 2. the final highlighted area after Translate fails to cover any part of the target; 3. the final highlighted area covers part or all of the target, but the size of the highlighted area is too large (refer to Figure 9.1 (b) as an example). We added the third condition because in real case an overly large highlighted area does not help the audience to understand
which is the target word, because there may be many words being highlighted. In this experiment, only highlighted area after six or seven times of Shrink (seven is the largest number allowed in the highlighting tool) were considered to be small enough for all targets. Error rate for each device type and target size combination was calculated respectively.

Figure 9.1: Examples of unsuccessful highlighting trials: (a). the highlighted area does not cover any part of the target; (b). the highlighted area covers the target, but it ends up being too large.

Second, we hypothesized that it would take longer time for the highlighting tool to complete the highlighting task, because the highlighting tool requires multiple clicks, while for a mouse only a translation and a click is required.

Task completion time was calculated only based on successful highlighting processes. For a mouse, it starts when the target appears on the screen, and ends when the user performs a click. For the highlighting tool, it also starts when the target appears on the screen, and ends when the user clicks a second ‘E’ key, which indicates the end of a trial.

We also had two hypotheses related to the difference of performance when highlighting different size of targets using the highlighting tool. First, we expected that target size would not affect error rate. It is true that when using a mouse it is more error-prone to select smaller targets [20]. However we assumed that because highlighting a Small target using the highlighting tool would require only one more Shrink step compared to a Large or a Medium target (namely there would be a total of seven Shrink steps, instead of six, which is usually the case for Large and Medium targets), which does not necessarily increase the error rate significantly. Second, we expected that there would be a significant effect of target size on completion time. More specifically, the smaller the target was, the slower the process would be. This hypothesis was based on the assumption that highlighting smaller targets requires one extra step of Shrink.

Finally we made four more hypotheses solely for the highlighting tool, which
we hoped could help us to better understand some details and features that were unique to this highlighting paradigm. First, we hypothesized that previous experience with remotes would not affect the highlighting speed. We assumed that the highlighting task was relatively easy for novice users and that they were able to achieve high speed without special training experience.

Second, we were interested in how participants held the remote when highlighting the target, especially how many hands were used. We expected that bi-manual interaction would be faster than single-handed interaction when using a remote.

Third, we define Coverage to be the percentage of the highlighted area inside the target (see Figure 9.2). Since it is usually impossible for the highlighted area and the target to overlap perfectly, this metric helps us understand the best extent to which they can overlap. We hypothesized that the smaller the target is, the lower the coverage would be.

![Figure 9.2: Black rectangle is the target, white area is the highlighted area, red area is the part of the highlighted area that is inside the target.](image)

Fourth, we would like to further look at the Shrink mode. We hypothesized that there would be a significant difference of per-click time among the sequence of buttons in Shrink mode, i.e. the time it took to click the 1st, 2nd, 3rd, . . . , 6th button in the Shrink mode was different. We expected that the time would increase
9.5 Results and Discussion

as the process went on. We assumed this was due to the fact that as the highlighted area became smaller, it was more likely that more than one quadruples would cover part of the target. Thus it would take longer time for participants to decide which quadruple contains the most portion of the target.

In summary, the goal of our evaluation is to test following hypotheses.

\( H1 \): Error rate will be the same for a remote and a mouse.
\( H2 \): Target size will not affect error rate.
\( H3 \): Highlighting with a remote will be slower than highlighting with a mouse.
\( H4 \): It takes more time to highlight smaller targets when using a remote.
\( H5 \): When using a remote, previous training experience does not make a difference in terms of highlighting speed.
\( H6 \): Bi-manual interaction will be more efficient than single-handed interaction.
\( H7 \): It is more difficult to perfectly highlight smaller targets.
\( H8 \): Per-click time in Shrink mode will increase as the Shrink process goes on.

9.5 Results and Discussion

We now present the experimental results and discussion. We first talk about error rate and highlighting time for both devices. We then solely look at the highlighting tool to analyze its usability.

9.5.1 Error Rate

We measured error rate for each trial with device and target size being independent variables. A repeated measure ANOVA showed that there was no significant effect of size on error rate \( (F_{1,97,37.3} = .14, p > .05, \eta^2_p = .007) \). Contrary to our anticipation, there was a significant effect of device \( (F_{1,19} = 14.5, p < .005, \eta^2_p = .43) \) on error rate. The interaction effect was not significant \( (F_{2,20,37.9} = .34, p > .05, \eta^2_p = .018) \). Figure 9.3 shows the average error rate for each device and size combination.

Conclusion

From these results, we rejected \( H1 \): error rate was higher when using a remote. However \( H2 \) was accepted: target size did not affect the error rate of highlighting. However, we would like to emphasize that while there was a significant effect on device, a discrepancy of about four percent is still acceptable.
9.5. Results and Discussion

9.5.2 Highlighting Time

The next dependent variable was time (in millisecond), the inverse of which was an indicator of speed, again with device and target size being independent variables. As anticipated, a repeated measure ANOVA revealed significant effect of device ($F_{1,19} = 226$, $p < .001$, $\eta^2_p = .92$) and size ($F_{1.80,34.3} = 14.5$, $p < .001$, $\eta^2_p = .43$) on highlighting time. The interaction was a significant factor as well ($F_{1.8,34.2} = 9.15$, $p < .005$, $\eta^2_p = .33$). Figure 9.4 depicts the average time used for highlighting one target. Simple main effect analysis showed that when using a mouse, there was a significant effect of target size ($F_{2,38} = 13.2$, $p < .001$, $\eta^2_p = .41$). Post-hoc analysis using Tukey’s method revealed that time spent on Small targets was significantly longer than that spent on Large ($q_{3.57} = 125$, $p < .001$) and Medium ($q_{3.57} = 91.5$, $p < .005$) ones, yet the difference between the time highlighting Large and Medium targets was not significant ($q_{3.57} = 33.4$, $p > .05$). Similar results were obtained when using a remote. Simple main effect analysis showed that there was also a significant effect of size ($F_{2,38} = 11.7$, $p < .001$, $\eta^2_p = .38$). Post-hoc analysis indicated that the difference was significant between large and small targets ($q_{3,57} = 1146$, $p < .005$), as well as between medium and small targets ($q_{3,57} = 1135$, $p < .005$). However it was not the case for large and medium targets ($q_{3,57} = 11.1$, $p > .05$).

Discussion

As mentioned previously the time spent highlighting a large or a medium target was not significantly different. This could be explained by the fact that to highlight
a large or a medium target, the highlighted area usually shrank to the same size; while for small targets participants would perform Shrink one more step, which led to longer highlighting time.

**Conclusion**

In conclusion the above results supported H3: highlighting with a remote was slower than highlighting with a mouse, as well as H4: it took more time to highlight small targets when using a remote.

![Figure 9.4: Highlighting time for each device and size combination.](image)

**9.5.3 Training Experience and Number of Hand Used**

Nine participants had used a clicker remote before. The average highlighting time for both experienced and inexperienced participants is depicted in Figure 9.5. Although it seemed that there was a difference in interaction time, repeated measure ANOVA showed otherwise: there was a significant effect only on size ($F_{1,35,10.8} = 10.1, p < .01, \eta^2_p = .56$), yet not on experience level ($F_{1,8} = .60, p > .05, \eta^2_p = .07$). Number of hands used was not a significant factor either ($F_{1,6} = .015, p > .05, \eta^2_p = .003$).

**Conclusion**

From the above results, we can conclude that H5 was supported: when using a remote to highlight a target, previous training experience did not make a difference
9.5. **Results and Discussion**

in terms of speed. However $H6$ was rejected: number of hand involved in the interaction does not affect highlighting speed.

![Figure 9.5: Interaction time for each experience level and size combination.](image)

**9.5.4 Coverage**

Average Coverage values for different target sizes are depicted in Figure 9.6. As expected, Coverage decreased as the size of the target became smaller. A repeated measure ANOVA revealed that there was a significant effect of size on Coverage ($F_{1,33,25.3} = 40.2, p < .001, \eta^2_p = .68$). Post-hoc analysis using Tukey’s method showed that Coverage for each size was significantly different from one another ($q_{3,57} = .054, p < .001$ for large and medium, $q_{3,57} = .20, p < .001$ for large and small, $q_{3,57} = .14, p < .001$ for medium and small).

**Conclusion**

In words, $H7$ was accepted: it is more difficult to perfectly highlight smaller targets.

**9.5.5 Per-click Time in Shrink Mode**

In Shrink mode, we compared the time used for the first, second, . . . , sixth click, and generate the plot, Figure 9.7 is obtained. Repeated measure ANOVA showed that there was a significant effect on index ($F_{2,62.49.7} = 40.3, p < .001, \eta^2_p = .68$), yet not on size ($F_{1,94.36.8} = 1.56, p > .05, \eta^2_p = .076$) and interaction ($F_{3,17,60.2} = 1.16, p > .05, \eta^2_p = .057$).
9.5. Results and Discussion

Figure 9.6: Coverage for different target sizes.

Discussion

Figure 9.7 showed that from the second click, the time it took to perform a click increased as the highlighted area getting smaller. This was probably because as the Shrink process went on, more than one quadruple would contain part of the target, in which case participants needed to determine which quadruple contained the most. However, the theory did not apply to the first click, which took much longer time than the second click. This was probably because when a new highlighting process started, the location of the target changed, and it took time for participant to figure it out where the new target was and how to perform the first Shrink.

Conclusion

The results above somewhat supported $H8$: Per-click time in Shrink mode will increase as the Shrink process goes on.

9.5.6 Results from Survey

We now present results obtained from the survey. Participants were asked if they knew any other way of using i>clicker besides multiple-choice question. One participant mentioned that he/she had seen i>clicker being used for roll call. Another participant mentioned using i>clicker remote to interact with a Gamelet, described in Chapter 5. When participants were asked if they had seen/heard of/experienced other ways of highlighting contents on a slide besides verbally mentioning its location, 2 participants mentioned using a laser pointer. Participants were then asked
to determine the usefulness of Separators and Labels when using the highlighting tool with five-point scales (1 = very obtrusive, 3 = neutral and 5 = very helpful), average scores of 4.35 and 4.25 were obtained for Separators and Labels respectively. It is worth noting that one participant said Labels were redundant because he/she was able to quickly memorize their locations in both modes after several trials of practicing.

At the end of the survey participants provided comments about the design of the highlighting tool. Six participants mentioned that they were not used to the layout of the Labels in the experiment, because the layout of buttons on the remote, which formed a straight line, did not correspond to the quadruple nature of the graphic interface. Four participants mentioned that during the Shrink mode, the position of Label C and D should be exchanged so that the four Labels could follow a traditional left to right, top to bottom fashion. Two participants mentioned that an “undo” functionality would be useful in the Shrink mode in case an error occurred. Two participants expected automatic Shrink technique so that users no longer needed to go through the long Shrink process, although they did not provide any insight regarding how the highlighting tool was able to know which word the user wanted to highlight.

9.6 Conclusion

In this section we first provide a conclusive summary of results obtained from this experiment. Compared to a mouse, the highlighting tool was significantly slower and more error-prone. However, we believed that in real case an error rate of
about 5% was still acceptable when using the highlighting tool. The discrepancy of highlighting time between two device types was larger. Participants spent about 16 seconds to highlight a target when using the highlighting tool, while it took less than 2 seconds when using a mouse. When using the highlighting tool, target size affected the highlighting time, but not error rate. Experimental results also showed that neither training experience nor number of hand affected highlighting time. Additionally, we found that it was more difficult to highlight smaller targets perfectly, and that it took more time for participants to make a decision of which quadruple to choose when the highlighting area became smaller.

We would like to further discuss the advantages and disadvantages of the highlighting tool in a real world scenario. It is true that the highlighting tool has a major disadvantage that it takes longer time than a mouse. However it is currently impossible for a student sitting in a class to highlight a target on the big screen with a mouse. The highlighting tool has another advantage over verbally mentioning the location of the target. In a big class, it is often very hard for students who sit at the back of the classroom to hear clearly what the question is asked by their classmates sitting at the front. The highlighting tool, however, is able to provide visual hints on the big screen to guide the audience to focus on the part that is in question on the slide.
Chapter 10

Conclusion and Future Work

The existing i>clicker SRS adopted by many universities is an excellent platform to build applications that supports classroom interaction and collaboration, mainly because there is no extra hardware infrastructure cost. To make use of the existing hardware, a driver working with the i>clicker base station was developed, which was able to fetch all the votes collected by the base station. With other necessary functionalities included, such as initializing the base station, start or stop voting, updating LCD, the driver provided a complete yet clean interface for developers to build various applications. WebClicker moves the traditional voting functionality to the cloud, and adds support to various digital devices using different services.

In this thesis, three applications were presented that support classroom interaction and collaboration in different ways. Clicker++ is a reproduced version of the i>clicker software, which introduces several new features, such as visualizing votes by lab group, student-oriented (instead of remote-oriented) identification, participation bar and performance bar. Clic^in is a pedagogical application that provides a way for students to practice newly-acquired knowledge right in the class, with both whole-class participation and group competition supported. Finally, we proposed a solution that enables a user to highlight contents on the slide using a remote.

Two laboratory studies were carried out and their results were discussed. The first study investigated the cognitive load of interacting with a gamelet using a remote. The result was compared with the ones using verbal commands, a mouse and a remote with color-coded interface respectively. Although it was found that a remote was significantly slower and more error-prone than a mouse, we argued that the discrepancy was acceptable in a real world scenario. It was also found that both interaction speed and accuracy, could be improved over time. The second study evaluates the usability of content highlighting feature of the Selection Tool, compared with a mouse. Experimental results revealed the same results. Using a remote was significantly slower and more error-prone than using a mouse. Also, when using a remote, size of the target affected interaction time, but no for error-rate.

There is considerable potential for future work that might be valuable.

First, the applications built based on the base station driver were just examples. More can be designed and implemented based on specific user scenarios or usabil-
ity problem discovered in a real class so that (1) it saves time for both instructors and students to complete a task which can be very tedious (roll call, for example), and (2) communication between instructors and students, instructors and teaching assistants, and among students can be more efficient and effective.

Second, both quantitative and qualitative studies can be carried out in a classroom that utilizes WebClicker to compare the usability of different digital devices in a classroom setting. Devices discussed in this thesis, e.g. laptops, tablets, smart phones and cell phones are inherently different in terms of size, weight and affordance, and it would be valuable to examine the effects of these differences on how students perform interaction in a classroom setting.

Third, currently Clic^in is designed for a computer science course. It is relatively easy for instructors or teaching assistants having programming background to write customized gamelets. However, this would be difficult for faculty members from other departments who do not have programming experience. One promising future direction might be investigating the probability of developing a gamelet authoring tool that can help instructors who do not have programming background to easily generate gamelet that is customized to their field. A good example is E-Prime [4], a psychology software tool that supports a drag and drop graphical interface for designing psychological experiment. We believe this improvement will remove the barrier between Clic^in and many potential users and will make the application more accessible and user friendly.
Bibliography


Appendices
Appendix A

Communication Protocol
Between i>clicker Base Station (Old) and PC

Packets sent between the base station and the PC can be described as an orthogonal combination of two attributes: function group and role. There are a total of five function groups: initializing base station, starting voting, requesting vote, stopping voting and updating LCD. There are three roles: A PC Command (PCC) is sent by the PC to the base station; a Base Station Acknowledgement (BSA) confirms to the base station that a PC command has been received; a Base Station Response (BSR) conveys useful information back to the base station about a PC command, e.g. the votes that the base station has collected. Table A.1 shows which packets exist for each function and role combination.

<table>
<thead>
<tr>
<th>Function Group</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>PC Command (PCC)</td>
</tr>
<tr>
<td>Initializing Base Station</td>
<td>Y</td>
</tr>
<tr>
<td>Starting Voting</td>
<td>Y</td>
</tr>
<tr>
<td>Requesting Vote</td>
<td>Y</td>
</tr>
<tr>
<td>Stopping Voting</td>
<td>Y</td>
</tr>
<tr>
<td>Updating LCD</td>
<td>Y</td>
</tr>
</tbody>
</table>

Table A.1: The Classification of packets for the old base station. Each packet has a role (PCC, BSA, or BSR) as indicated in the columns, and a function group, as indicated in the rows.

Each packet is described in detail below according its function group, on function group per sub-section. The role of the packet is indicated in the table caption, with a number differentiating among packets belonging to the same role group (e.g.
PCC 1, PCC 2, etc.). All table cells, except the last row, constitute the data area, with hex numbers representing the data in the packet. Sometimes labels are used for the data (e.g. InstructorID), which are described in the last row of the table, which is a brief comment is either explaining what a label means or providing extra information to help readers to understand the packet. PC Commands are labeled in sequence, e.g. PCC 1, PCC 2, etc. Base Station Acknowledgement and Base Station Response are labeled according to the corresponding PC Command, e.g. BSA 5 and BSR 5 are the acknowledgement and response packets, respectively, for PCC 5.

The protocol for the old base station was determined by reverse engineering, which involved using the base station with the vendor-provided software and “sniffing” the protocol to determine the data that was being transmitted between the base station and the PC on the USB port. The descriptions that follow are the result of that exercise. There is no guarantee that the list of commands is complete (the list includes all of the commands that we observed) nor that the interpretation of the commands is accurate (it is consistent with the understanding we developed through the reverse engineering exercise, but it may be incomplete in details). Moreover, because we switched out development to the new base station when it became available, we did not fully explore the protocol for the old base station beyond what we needed for our driver.

### A.1 Initializing Base Station

The following packets are transmitted when the PC initializes the base station.

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x10</td>
<td>BF1</td>
<td>BF2</td>
</tr>
<tr>
<td>. . .</td>
<td>(25 words (96 bytes), omitted, all being 0x00) . . .</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This packet is sent every time PC initializes the base station. Order of BF1 and BF2 cannot be changed.

- BF1: Base frequency 1, ranging from 0x21 to 0x24.
- BF2: Base frequency 2, ranging from 0x41 to 0x44.

Table A.2: PCC 1 - Set the frequency of the base station
A.1. Initializing Base Station

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x17</td>
<td>0x06</td>
<td>InstructorID0</td>
</tr>
<tr>
<td>InstructorID1</td>
<td>2</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (14 words (56 bytes), omitted, all being 0x00) ... 

This packet is sent every time PC initializes the base station, if instructor’s clicker ID is provided.

InstructorID: Instructor’s clicker ID, first six characters only. Only one remote can be set as an instructor’s clicker, which can be any remote (doesn’t have to be the blue one).

Table A.3: PCC 2 - Set instructor’s remote ID

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x16</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (15 words (60 bytes), omitted, all being 0x00) ... 

This packet is sent every time PC initializes the base station. This packet tells the base station to switch to “not-accepting” mode.

Table A.4: PCC 3

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x16</td>
<td>0xAA</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (15 words (60 bytes), omitted, all being 0x00) ... 

This packet is sent as an acknowledgement of PCC 3, only if the base station receives PCC 2.

Table A.5: BSA 3
A.2 Starting Voting

The following packets are transmitted when PC starts voting.

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x17</td>
<td>0x03</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (15 words (60 bytes), omitted, all being 0x00) ...

This packet is sent every time PC initializes the base station.

Table A.6: PCC 4

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x17</td>
<td>0xAA</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (15 words (60 bytes), omitted, all being 0x00) ...

This packet is sent as an acknowledgement of PCC 4.

Table A.7: BSA 4

A.2 Starting Voting

The following packets are transmitted when PC starts voting.

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x17</td>
<td>0x05</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (15 words (60 bytes), omitted, all being 0x00) ...

This packet is sent every time PC starts voting.

This packet reset the base station.

Table A.8: PCC 5
A.3. Requesting Vote

The following packets are transmitted when PC requests fresh votes.

Table A.9: BSA 5

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x17</td>
<td>0xAA</td>
<td>0x00</td>
</tr>
</tbody>
</table>

This packet is sent as an acknowledgement of PCC 5.

Table A.10: PCC 6

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x11</td>
<td>0x00</td>
<td>0x05</td>
</tr>
</tbody>
</table>

This packet is sent every time PC starts voting. This packet tells the base station to switch to “accepting” mode.

Table A.11: BSA 6

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x11</td>
<td>0xAA</td>
<td>0x00</td>
</tr>
</tbody>
</table>

This packet is be sent as an acknowledgement of PCC 6.
A.3. Requesting Vote

This packet asks base station to send fresh votes, as long as the base station connection is opened. It is sent every 0.1 second in the i>clicker software.

LV: Index of last received vote. It tells the base station that PC has received all the votes up to LV, and please send LV+1, LV+2, ..., etc, if any, to PC. If LV is zero, it means PC has not received any vote yet. LV is only one byte, whose value ranges from 0 to 255. However, it is found that the base station can only send as many as 56 votes in one response set.

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x17</td>
<td>0x01</td>
<td>LV</td>
</tr>
<tr>
<td>0x04</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

. . . (14 words (56 bytes), omitted, all being 0x00) . . .

Table A.12: PCC 7 - Request new votes from the base station

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x17</td>
<td>0xAA</td>
<td>0x00</td>
</tr>
</tbody>
</table>

. . . (15 words (60 bytes), omitted, all being 0x00) . . .

This packet is sent as an acknowledgement of PCC 7.

Table A.13: BSA 7
### A.3. Requesting Vote

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>0x18</td>
<td>LastFlag</td>
<td>VoteLeft</td>
</tr>
<tr>
<td>LVL</td>
<td>LVC</td>
<td>VoteCurrent</td>
<td>Choice1</td>
</tr>
<tr>
<td>ClickerID1</td>
<td></td>
<td></td>
<td>Choice2</td>
</tr>
<tr>
<td>ClickerID2</td>
<td></td>
<td></td>
<td>Choice3</td>
</tr>
<tr>
<td>ClickerID3</td>
<td></td>
<td></td>
<td>. . . (11 words (44 bytes), omitted) . . .</td>
</tr>
<tr>
<td>. . . (11 words (44 bytes), omitted) . . .</td>
<td></td>
<td></td>
<td>CheckSum</td>
</tr>
</tbody>
</table>

This packet is sent after BSA 7, as a follow-up response of PCC 7. It contains all the fresh votes.

**LastFlag**: A flag indicating whether this is the last response packet of the current response set. One BSR 7 packet can hold a maximum of 14 votes. If between two adjacent polls (PCC 7 packets), more than 14 new votes come in, more than one BSR 7 packet will be generated in order to send all the new votes. LastFlag of 0x01 tells PC this packet is not the last packet of the current response set. Instead, LastFlag with a value of 0x00 indicates that this is the last one. For example, if between two adjacent polls, 38 new votes come in, the base station will generate 3 BSR 7 packets, with LastFlag being 0x01, 0x01, 0x00, each holding 14, 14 and 10 votes respectively.

**VoteLeft**: Total amount of votes left in this response set, namely, the total amount of new votes that are received by the base station, but not yet sent to PC. In the above example, VoteLeft will be 38 for the first packet, 24 for the second packet and 10 for the last one.

**LVL**: The vote that has the largest vote index in the last packet. LVL+1 is the index of the first vote in the current packet.

**LVC**: The vote that has the largest vote index in the current packet.

**VoteCurrent**: Total amount of vote(s) in the current packet. VoteCurrent = LVC - LVL.

**Choice{1,2,...,14}**: Choice of the {first, second, ..., fourteenth} vote in the current packet, 0x61 for A, 0x62 for B, 0x63 for C, 0x64 for D and 0x65 for E.

**ClickerID{1,2,...,14}**: Clicker ID of the {first, second, ..., fourteenth} vote in the current packet, first six characters only.

**CheckSum**: Checksum of the current packet.

| Table A.14: BSR 7 - New votes from the base station |
A.4 Stopping Voting

The following packets are transmitted when PC stops voting.

Table A.15: PCC 8

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x16</td>
<td>0xAA</td>
<td>0x00</td>
</tr>
</tbody>
</table>

. . . (15 words (60 bytes), omitted, all being 0x00) . . .

This packet is sent as an acknowledgement of PCC 8.

Table A.16: BSA 8

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x17</td>
<td>0x04</td>
<td>0x00</td>
</tr>
</tbody>
</table>

. . . (15 words (60 bytes), omitted, all being 0x00) . . .

This packet is sent every time PC stops voting.
This packet requests the summary of the current voting session.

Table A.17: PCC 9
A.5. Updating LCD

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x17</td>
<td>0xAA</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (15 words (60 bytes), omitted, all being 0x00) ...

This packet is sent as an acknowledgement of PCC 9.

Table A.18: BSA 9

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>0x18</td>
<td>0x16</td>
<td>0x00</td>
</tr>
<tr>
<td>?</td>
<td>0x00</td>
<td>?</td>
<td>0x00</td>
</tr>
<tr>
<td>0x00</td>
<td>0x00</td>
<td>?</td>
<td>0x00</td>
</tr>
<tr>
<td>TotalVote</td>
<td>0x00</td>
<td>0x01</td>
<td>0x00</td>
</tr>
<tr>
<td>0x16</td>
<td>0x00</td>
<td>TotalVote</td>
<td>0x00</td>
</tr>
<tr>
<td>InstructorID</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (9 words (36 bytes), omitted, all being 0x00) ...

This packet is sent after BSA 9, as a follow-up response of PCC 9. It provides a summary of the current voting session.

TotalVote: Total amount of votes. As long as a vote is received by the base station, it counts, no matter whether the person who contributes this vote has voted earlier in this vote session, or who, either an instructor or a student, contributes this vote.

InstructorID: Instructor’s clicker ID, first six characters only.

Table A.19: BSR 9 - Vote count summary from base station

A.5 Updating LCD

The following packets are transmitted when content displayed on the LCD is updated.
A.5. Updating LCD

This packet is sent every time the first line of the base station LCD needs to be updated. $C_x$: ASCII code of a character. The first line of the base station LCD can hold a maximum of 16 characters.

Table A.20: PCC 10 - Set first line of base station LCD

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x13</td>
<td>$C_0$</td>
<td>$C_1$</td>
</tr>
<tr>
<td>$C_2$</td>
<td>$C_3$</td>
<td>$C_4$</td>
<td>$C_5$</td>
</tr>
<tr>
<td>$C_6$</td>
<td>$C_7$</td>
<td>$C_8$</td>
<td>$C_9$</td>
</tr>
<tr>
<td>$C_{10}$</td>
<td>$C_{11}$</td>
<td>$C_{12}$</td>
<td>$C_{13}$</td>
</tr>
<tr>
<td>$C_{14}$</td>
<td>$C_{15}$</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (11 words (44 bytes), omitted, all being 0x00) ...

This packet is sent every time the second line of the base station LCD needs to be updated. $C_x$: ASCII code of a character. The second line of the base station LCD can hold a maximum of 16 characters.

Table A.21: PCC 11 - Set second line of base station LCD

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x14</td>
<td>$C_0$</td>
<td>$C_1$</td>
</tr>
<tr>
<td>$C_2$</td>
<td>$C_3$</td>
<td>$C_4$</td>
<td>$C_5$</td>
</tr>
<tr>
<td>$C_6$</td>
<td>$C_7$</td>
<td>$C_8$</td>
<td>$C_9$</td>
</tr>
<tr>
<td>$C_{10}$</td>
<td>$C_{11}$</td>
<td>$C_{12}$</td>
<td>$C_{13}$</td>
</tr>
<tr>
<td>$C_{14}$</td>
<td>$C_{15}$</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (11 words (44 bytes), omitted, all being 0x00) ...
Appendix B

Communication Protocol Between i>clicker Base Station (New) and PC

The protocol for the new base station is described in a manner similar to how the protocol for the old base station was described. Table B.1, a classification of packets based on function group and role group, is provided for the ease of reference, although it is exactly the same as Table A.1

<table>
<thead>
<tr>
<th>Function Group</th>
<th>PC Command (PCC)</th>
<th>Base Station Acknowledgement (BSA)</th>
<th>Base Station Response (BSR)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initializing Base Station</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Starting Voting</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Requesting Vote</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Stopping Voting</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Updating LCD</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

Table B.1: The Classification of packets for the new base station. Each packet has a role (PCC, BSA, or BSR) as indicated in the columns, and a function group, as indicated in the rows.

The protocol for the new base station is more complex. As was shown in Table A.1, the format of all the Base Station Acknowledgement packets (BSA) has a pattern. The first two bytes are always the same as the first two bytes of the corresponding PC Command packets (PCC), the third byte is always 0xAA, and the rest of the bytes are all 0x00. Thus one can tell what a BSA packet looks like given the description of its corresponding PCC packet. Thus, for the sake of conciseness, in the protocol of the new base station, all the BSA packets are omitted, except for the first example (PCC 1), but the comment area of the corresponding PCC packet
B.1 Initializing Base Station

indicates the BSA that is expected.

The protocol for the new base station was determined by reverse engineering, which involved using the base station with the vendor-provided software and “sniffing” the protocol to determine the data that was being transmitted between the base station and the PC on the USB port. The descriptions that follow are the result of that exercise. There is no guarantee that the list of commands is complete (the list includes all of the commands that we observed) nor that the interpretation of the commands is accurate (it is consistent with the understanding we developed through the reverse engineering exercise, but it may be incomplete in details).

B.1 Initializing Base Station

The following packets are transmitted when the PC initializes the base station.

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x10</td>
<td>BF1</td>
<td>BF2</td>
</tr>
</tbody>
</table>

... (15 words (60 bytes), omitted, all being 0x00) ...

This packet is sent every time PC initializes the base station.
The order of BF1 and BF2 cannot be changed.
BF1: Base frequency 1, ranging from 0x21 to 0x24.
BF2: Base frequency 2, ranging from 0x41 to 0x44.

Table B.2: PCC 1

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x10</td>
<td>0xAA</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (15 words (60 bytes), omitted, all being 0x00) ...

This acknowledgement packet is sent to the PC every time a PCC 1 is received by the base station.

Table B.3: BSA 1
### B.1. Initializing Base Station

This packet is sent every time PC initializes the base station.

#### BF1: Base frequency 1, ranging from 0x21 to 0x24.

#### BF2: Base frequency 2, ranging from 0x41 to 0x44.

#### Table B.4: PCC 2

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x12</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

. . . (15 words (60 bytes), omitted, all being 0x00) . . .

This packet is sent every time PC initializes the base station.

An acknowledgement packet BSA 3 is returned.

#### Table B.5: PCC 3

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x16</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

. . . (15 words (60 bytes), omitted, all being 0x00) . . .

This packet is sent every time PC initializes the base station.

An acknowledgement packet BSA 4 is returned.

#### Table B.6: PCC 4
B.1. Initializing Base Station

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x1E</td>
<td>InstructorID₀</td>
<td>InstructorID₁</td>
</tr>
</tbody>
</table>

InstructorID₂ 0x00 0x00 0x00

... (15 words (60 bytes), omitted, all being 0x00) ...

This packet is sent every time PC initializes the base station, if instructor’s clicker ID is provided.
InstructorID: Instructor’s clicker ID, first six characters only. Only one remote can be set as an instructor’s clicker, which can be any remote (doesn’t have to be the blue one).
An acknowledgement packet BSA 5 is returned.

Table B.7: PCC 5 - Set instructor’s remote ID

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x15</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (15 words (60 bytes), omitted, all being 0x00) ...

This packet is sent every time PC initializes the base station.

Table B.8: PCC 6
B.1. Initializing Base Station

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x15</td>
<td>V1</td>
<td>V2</td>
</tr>
<tr>
<td>BF1</td>
<td>BF2</td>
<td>0x01</td>
<td>0x02</td>
</tr>
<tr>
<td>0x66</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (13 words (52 bytes), omitted, all being 0x00) ...

This packet is sent as a response to PPC 6, which contains base station information, like firmware version and frequency used.
V1: Version number 1, major version of the base station; 02 for the old one and 04 for the new one.
V2: Version number 2, minor version of the base station; 03 for the old one and 05 for the new one.
BF1: Base frequency 1 currently used by the base station, ranging from 0x21 to 0x24.
BF2: Base frequency 2 currently used by the base station, ranging from 0x41 to 0x44.

Table B.9: BSR 6 - Base station firmware and frequency

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>0x2C</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (15 words (60 bytes), omitted, all being 0x00) ...

This packet is sent as a response of nothing (seems to be). Sometimes the packet appears here, other times it is after PPC 2 or BSA 6 or BSR7.

Table B.10: BSR X

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x2D</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (15 words (60 bytes), omitted, all being 0x00) ...

This packet is sent every time PC initializes the base station.
An acknowledgement packet BSA 7 is returned.

Table B.11: PCC 7
### B.1. Initializing Base Station

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x29</td>
<td>0xA1</td>
<td>0x8F</td>
</tr>
<tr>
<td>0x96</td>
<td>0x8D</td>
<td>0x99</td>
<td>0x97</td>
</tr>
<tr>
<td>0x8F</td>
<td>0x80</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (13 words (52 bytes), omitted, all being 0x00)...

This packet is sent every time PC initializes the base station. An acknowledgement packet BSA 8 is returned.

**Table B.12: PCC 8**

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x17</td>
<td>0x04</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (15 words (60 bytes), omitted, all being 0x00)...

This packet is sent every time PC initializes the base station. An acknowledgement packet BSA 9 is returned.

**Table B.13: PCC 9**
B.1. Initializing Base Station

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>0x18</td>
<td>0x1A</td>
<td>TotalVote_0</td>
</tr>
<tr>
<td>TotalVote_1</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>0x00</td>
<td>TotalVote</td>
<td>TotalVote_0</td>
<td></td>
</tr>
<tr>
<td>TotalVote_1</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x01</td>
</tr>
<tr>
<td>InstructorID</td>
<td>0x00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x01</td>
<td>0x00</td>
<td>V1</td>
<td>v2</td>
</tr>
</tbody>
</table>

... (9 words (36 bytes), omitted, all being 0x00)...

This packet is sent as a response to PCC 9, which provides a summary of the previous voting session.

TotalVote: Total amount of votes received in the previous question. As long as a vote is received by the base station, it counts, no matter whether the person who contributes this vote has voted earlier in this vote session, or who, either an instructor or a student, contributes this vote. This field occupies two bytes.

InstructorID: Instructor’s clicker ID, first six characters only.

V1: Version number 1, major version of the base station; 02 for the old one and 04 for the new one.

V2: Version number 2, minor version of the base station; 03 for the old one and 05 for the new one.

Table B.14: BSR 9 - Summary of the previous voting session

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x17</td>
<td>0x03</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (15 words (60 bytes), omitted, all being 0x00)... This packet is sent every time PC initializes the base station.

An acknowledgement packet BSA 10 is returned.

Table B.15: PCC 10

100
B.2. Starting Voting

Table B.16: PCC 11

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x16</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

. . . (15 words (60 bytes), omitted, all being 0x00) . . .

This packet is sent every time PC initializes the base station. An acknowledgement packet BSA 11 is returned.

Table B.17: PCC 12

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x19</td>
<td>0x66</td>
<td>0x0A</td>
</tr>
<tr>
<td>0x01</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

. . . (14 words (56 bytes), omitted, all being 0x00) . . .

This packet is sent every time PC starts voting. An acknowledgement packet BSA 12 is returned.

Table B.18: PCC 13

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x17</td>
<td>0x03</td>
<td>0x00</td>
</tr>
</tbody>
</table>

. . . (15 words (60 bytes), omitted, all being 0x00) . . .

This packet is sent every time PC starts voting. An acknowledgement packet BSA 13 is returned.
B.3. Requesting Vote

This packet is sent every time PC starts voting.
An acknowledgement packet BSA 14 is returned.

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x17</td>
<td>0x05</td>
<td>0x00</td>
</tr>
</tbody>
</table>

. . . (15 words (60 bytes), omitted, all being 0x00) . . .

Table B.19: PCC 14

This packet is sent every time PC starts voting.
An acknowledgement packet BSA 15 is returned.

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x19</td>
<td>0x66</td>
<td>0x0A</td>
</tr>
<tr>
<td>0x01</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

. . . (14 words (56 bytes), omitted, all being 0x00) . . .

Table B.20: PCC 15

This packet is sent every time PC starts voting.
An acknowledgement packet BSA 16 is returned.

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x11</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

. . . (15 words (60 bytes), omitted, all being 0x00) . . .

Table B.21: PCC 16

B.3 Requesting Vote

The following packets are transmitted when PC requests fresh votes.
### B.3. Requesting Vote

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x11</td>
<td>0xAA</td>
<td>0x00</td>
</tr>
</tbody>
</table>

. . . (7 words (28 bytes), omitted, all being 0x00) . . .

<table>
<thead>
<tr>
<th>0x02</th>
<th>0x13</th>
<th>Choice</th>
<th>ClickerID</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>0x81</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x82</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x83</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x84</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x85</td>
<td></td>
</tr>
</tbody>
</table>

. . . (6 words (24 bytes), omitted, all being 0x00) . . .

Once the base station receives fresh votes, it will automatically send the votes to PC. This packet is sent when the vote in this packet is the first vote in the current voting session.

**Choice:** Choice of the vote in the current packet, 0x81 for A, 0x82 for B, 0x83 for C, 0x84 for D and 0x85 for E.

**ClickerID:** Clicker ID of the vote in the current packet, first six characters only.

**VoteIndex:** Index of the vote in the current packet. When a new question starts, the index continues, instead of starting from zero.

Table B.22: BSR Y - First new vote from the base station
B.4 Stopping Voting

The following packets are transmitted when PC stops voting.

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x12</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (15 words (60 bytes), omitted, all being 0x00) ...

This packet is sent every time PC stops voting.

Table B.24: PCC 17
### B.4. Stopping Voting

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>0x12</td>
<td>0xAA</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (7 words (28 bytes), omitted, all being 0x00) ...

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Choice</th>
<th>ClickerID</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>0x13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ClickerID</td>
<td>VoteIndex</td>
<td>0x00</td>
<td></td>
</tr>
</tbody>
</table>

... (6 words (24 bytes), omitted, all being 0x00) ...

This packet is sent as a response to PCC 17.
Choice: Choice of last vote received.
ClickerID: Clicker ID of the vote in the current packet, first six characters only.
VoteIndex: Vote index of the last vote received.

Table B.25: BSR 17

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x16</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (15 words (60 bytes), omitted, all being 0x00) ...

This packet is sent every time PC stops voting.
An acknowledgement packet BSA 18 is returned.

Table B.26: PCC 18

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x17</td>
<td>0x01</td>
<td>0x00</td>
</tr>
<tr>
<td>0x04</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (14 words (56 bytes), omitted, all being 0x00) ...

This packet is sent every time PC stops voting.
An acknowledgement packet BSA 19 is returned.

Table B.27: PCC 19

Another PCC 19 goes here.
### B.4. Stopping Voting

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x17</td>
<td>0x03</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (15 words (60 bytes), omitted, all being 0x00) ...

This packet is sent every time PC stops voting.
An acknowledgement packet BSA 20 is returned.

**Table B.28: PCC 20**

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x17</td>
<td>0x04</td>
<td>0x00</td>
</tr>
</tbody>
</table>

... (15 words (60 bytes), omitted, all being 0x00) ...

This packet is sent every time PC stops voting.
An acknowledgement packet BSA 21 is returned.

**Table B.29: PCC 21**
B.5. Updating LCD

This packet is sent as a response to PCC 21, which provides a summary of the current voting session.

TotalVote: Total amount of votes received in the current question. As long as a vote is received by the base station, it counts, no matter whether the person who contributes this vote has voted earlier in this vote session, or who, either an instructor or a student, contributes this vote. This field occupies two bytes.

Instructor’s clicker ID, first six characters only.

V1: Version number 1, major version of the base station; 02 for the old one and 04 for the new one.

V2: Version number 2, minor version of the base station; 03 for the old one and 05 for the new one.

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x02</td>
<td>0x18</td>
<td>0x1A</td>
<td>TotalVote0</td>
</tr>
<tr>
<td>TotalVote1</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>0x00</td>
<td>TotalVote</td>
<td>0x00</td>
<td>TotalVote0</td>
</tr>
<tr>
<td>TotalVote1</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
<td>0x00</td>
</tr>
<tr>
<td>InstructorID</td>
<td>0x00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0x01</td>
<td>0x00</td>
<td>V1</td>
<td>V2</td>
</tr>
</tbody>
</table>

. . . (9 words (36 bytes), omitted, all being 0x00) . . .

Table B.30: BSR 21 - Vote count summary from base station

B.5 Updating LCD

The following packets are transmitted when content displayed on the LCD is updated.
### B.5. Updating LCD

This packet is sent every time the first line of the base station LCD needs to be updated.

\( C_x \): ASCII code of a character. The first line of the base station LCD can hold a maximum of 16 characters.

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x13</td>
<td>( C_0 )</td>
<td>( C_1 )</td>
</tr>
<tr>
<td>( C_2 )</td>
<td>( C_3 )</td>
<td>( C_4 )</td>
<td>( C_5 )</td>
</tr>
<tr>
<td>( C_6 )</td>
<td>( C_7 )</td>
<td>( C_8 )</td>
<td>( C_9 )</td>
</tr>
<tr>
<td>( C_{10} )</td>
<td>( C_{11} )</td>
<td>( C_{12} )</td>
<td>( C_{13} )</td>
</tr>
<tr>
<td>( C_{14} )</td>
<td>( C_{15} )</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

. . . (11 words (44 bytes), omitted, all being 0x00) . . .

Table B.31: PCC 22 - Set first line of base station LCD

This packet is sent every time the second line of the base station LCD needs to be updated.

\( C_x \): ASCII code of a character. The second line of the base station LCD can hold a maximum of 16 characters.

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x01</td>
<td>0x14</td>
<td>( C_0 )</td>
<td>( C_1 )</td>
</tr>
<tr>
<td>( C_2 )</td>
<td>( C_3 )</td>
<td>( C_4 )</td>
<td>( C_5 )</td>
</tr>
<tr>
<td>( C_6 )</td>
<td>( C_7 )</td>
<td>( C_8 )</td>
<td>( C_9 )</td>
</tr>
<tr>
<td>( C_{10} )</td>
<td>( C_{11} )</td>
<td>( C_{12} )</td>
<td>( C_{13} )</td>
</tr>
<tr>
<td>( C_{14} )</td>
<td>( C_{15} )</td>
<td>0x00</td>
<td>0x00</td>
</tr>
</tbody>
</table>

. . . (11 words (44 bytes), omitted, all being 0x00) . . .

Table B.32: PCC 23 - Set second line of base station LCD
Appendix C

Participant Survey: Comparison of the Cognitive Load of Different Gamelet Interaction Techniques

Each participant in the laboratory study described in Chapter 7 was asked to complete the following survey questionnaire that was designed to elicit information about the participant’s experience using the i>clicker system and the cognitive load of the interaction techniques tested in the study.
Participant Survey: Comparison the Cognitive Load of Different Gamelet Interaction Techniques

Part I (Background Information):

1. Gender

Male                Female

2. Please specify if you are a in

High School  1st Year Undergrad  2nd Year Undergrad  3rd Year Undergrad

4th Year Undergrad  5th or Higher Undergrad  Master  PhD

Other, please specify: ______________________________________________________________

3. If you are receiving post-secondary education, please specify your major area.

Applied Science  Arts  Commerce and Business Administration  Dentistry  Education

Forestry  Land and Food Systems  Law  Medicine  Science

Other, please specify: ______________________________________________________________

Part II (Survey on i-clicker): In this part, you will be asked some questions about how i-clicker system was used in your past education experience.

4. Have you ever used i-clicker before?

Yes                No
5. How many courses did you take for Winter Term 1, 2012?
_________________________________________

Among these courses, how many use i>clicker?
______________________________________________

6. How many courses did you take for Winter Term 2, 2012?
_________________________________________

Among these courses, how many use i>clicker?
______________________________________________

Part III (General Opinion towards Gamelet): In this part, you will be asked some
general questions about Gamelet.

7. Have you ever seen/heard of/experienced other ways of using i>clickers in the class besides classic multiple choice question?

Yes
No

If Yes, please specify: ______________________________________________________

8. Assume that you are a student in a class where the instructor is explaining binary search tree. To be specific, the instructor is talking about how to insert a new node into the current binary search tree. Please indicate your attitude of the instructor using Gamelet in the class while explaining the idea.

Strongly Negative  Mildly Negative  Neutral  Mildly Positive  Strongly Positive

9. If you choose “Mildly Positive” or “Strongly Positive” in Question 8, find out two most important criteria that affects your decision.

☐ Easy access to the device (most students already own an i>clicker remote and they bring them to school anyway; not need to buy extra device).

☐ Increase of classroom participation and sense of involvement (I feel I would participate more in this type of classroom activity).
Combination of learning and entertainment (this game-type of application is more fun than classic way of teaching: students listen while the teacher talks).

Visibility (I can see and understand the result of my interaction on the big screen).

Other, please specify: ________________________________

10. If you choose “Mildly Negative” or “Strongly Negative” in Question 8, find out two most important criteria that affects your decision.

- I feel none the designs of the Gamelet are good enough for me to practice binary search tree and to understand how inserting a new node works.

- I feel the demonstration and practice of Gamelet takes too much time, which can be better used to cover material that is more useful.

Other, please specify: ________________________________

Part IV (Follow-up Interview): In this part, you will be interviewed for a few questions.

- For the best design of Gamelet, do you think is there anything needs to be added/deleted/changed so that the design gets even better? Any other comments regarding to Gamelet in general is also welcome.

________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________

Thank you so much for your participation!
Appendix D

Participant Survey: Comparison the Cognitive Load of Different Gamelet Interaction Technique (Follow-up Study)

Each participant in the follow-up laboratory study described in Chapter 7 was asked to complete the following survey questionnaire that was designed to elicit information about the participant’s experience using the i>clicker system.
Participant Survey: Comparison the Cognitive Load of Different Gamelet Interaction Techniques (Follow-up Study)

Part I (Background Information):

1. Gender

   Male                      Female

2. Please specify if you are a in

   High School       1st Year Undergrad   2nd Year Undergrad   3rd Year Undergrad

   4th Year Undergrad   5th or Higher Undergrad   Master   PhD

   Other, please specify: ____________________________________________________________

3. If you are receiving post-secondary education, please specify your major area.

   Applied Science   Arts   Commerce and Business Administration   Dentistry   Education

   Forestry   Land and Food Systems   Law   Medicine   Science

   Other, please specify: ____________________________________________________________

Part II (Survey on i>clicker): In this part, you will be asked some questions about how i>clicker system was used in your past education experience.

4. Have you ever used i>clicker before?

   Yes                      No
5. How many courses did you take for Winter Term 1, 2012?

_________________________________________

Among these courses, how many use i>clicker?

_________________________________________

6. How many courses did you take for Winter Term 2, 2012?

_________________________________________

Among these courses, how many use i>clicker?

_________________________________________

Thank you so much for your participation!
Appendix E

Participant Survey: Comparing the Performance of Highlighting using i>clicker Remote and Mouse

Each participant in the laboratory study described in Chapter 9 was asked to complete the following survey questionnaire that was designed to elicit information about about the participant’s experience using the i>clicker system and opinions about the interaction techniques for highlighting that were tested in the study.
Participant Survey: Comparing the Performance of Highlighting using i>clicker Remote and Mouse

Part I (Background Information):

1. Gender

Male
Female

2. Please specify if you are a in

High School
1st Year Undergrad
2nd Year Undergrad
3rd Year Undergrad

4th Year Undergrad
5th or Higher Undergrad
Master
PhD

Other, please specify: ______________________________________________________________

3. If you are receiving post-secondary education, please specify your major area.

Applied Science
Arts
Commerce and Business Administration
Dentistry
Education
Forestry
Land and Food Systems
Law
Medicine
Science

Other, please specify: ______________________________________________________________

Part II (Survey on i>clicker): In this part, you will be asked some questions about how i>clicker system was used in your past education experience.

4. Have you ever used i>clicker before?

Yes
No
5. How many courses did you take for Winter Term 1, 2013?

______________________________________________

Among these courses, how many used i>clicker?

______________________________________________

6. How many courses did you take for Winter Term 2, 2013?

______________________________________________

Among these courses, how many used i>clicker?

______________________________________________

**Part III (General Opinion towards Selection Tool):** In this part, you will be asked some general questions about Selection Tool.

7. Have you ever seen\heard of\experienced other ways of using i>clickers in the class besides classic multiple choice question?

Yes

No

If Yes, please specify: ____________________________________________________________

8. Assume that in a class, a student has a question about the definition of a word displayed on the slide, and s/he wants to refer to this word when asking the question. Have you ever seen\heard of\experienced other ways of selecting or highlighting content (e.g. words, symbols, cells in tables) on the slide in the class besides verbally mentioning where it locates on the slide?

Yes

No

If Yes, please specify: ____________________________________________________________

**Part IV (Design of Selection Tool):** In this part, you will be asked some questions about your opinion towards the visualization design of the Selection Tool (i.e. we will not consider mouse for this part). There are two different visual aids, Separator and Label.
9. What do you think of the Separator when making the decision regarding to which sub-area to pick?

Very Obstructive  Obstructive  Neutral  Helpful  Very Helpful

10. What do you think of the Label when making the decision regarding to which sub-area to pick?

Very Obstructive  Obstructive  Neutral  Helpful  Very Helpful

Part V (Follow-up Interview): In this part, you will be interviewed for a few questions.

- For the best design using i>clicker remote, do you think is there anything needs to be added\deleted\changed so that the design gets even better?

____________________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________
____________________________________________________________________________

Thank you so much for your participation!