Custom Keys

By

Cameron Hom

June 4, 2015

Table of Contents

Chapter 1: Introduction

In this chapter, I address the problem of user inefficient numeric input for most modern design laptops and propose a solution for the problem.

1.1 The Problem: Inefficient User Numeric Input

The number pad is structured for efficient number input . However, an increasing number of laptops do not have a number pad included. This is because most laptops designs are built to be compact in size and light in weight. For an example, the ASUS Zenbook UX305, is an ultra lightweight laptop with a sleek design (*ASUS, 2*). Achieving this type of design also means sacrificing other implementations such as the number pad on standard QWERTY keyboards. This design is unfortunate for the users who like the numeric pad.

There are current solutions that approach this problem. The first solution is the use of an external number pad device that can be connected through a Universal Serial Bus (USB). The user then can use the external numeric keypad like a normal keypad. The second solution is holding down a special function key, referred to the “fn” key, is usually included on laptop keyboards. This key changes the functionality of other existing keys into keys, which can emulate a similar pad design compared to the standard number pad. However, these previous approaches for solving people problems with numeric input can be solved in simple way. This problem can be solved with an app on a smartphone called Custom Keys.

1.2 Proposed Idea: Custom Keys

Custom Keys is a smart phone application that gives the user the functionality to input specific keystrokes directly from their phone to their computer. The keystroke data is sent to the computer through a wireless method involving a client and server. Custom Keys currently has one key configuration, the numeric keypad. This configuration displays a keypad similar to standard computer keyboards. I envision that this

Figure 1. ASUS Zenbook UX305 with Samsung Galaxy S4 running Custom Keys app side by side. May, 2015. Cameron Hom

app will be used alongside a laptop so users can have access to a numeric pad wherever they travel (Figure 1).

Chapter 2: Background Research

In this chapter, I elaborate on my research related to the problem and solution of inefficient user numeric input. I also talk about different methods of computer communication, the advantages and disadvantages between each method, and Android application development research.

2.1 Accessibility of Custom Keys

Smartphones are prevalent in modern society. According to Pew Research Center, 85% of individuals, ranging from the ages of 18 to 29, own a smartphone in the United States (*Pew Research, 1*). Custom Keys is a utility app which makes people's daily lives easier by improving numeric input tasks on laptops. Since most young adults in the United States usually carry smart phones in their pockets, Custom Keys will be an easily accessible app that individuals can utilize into their lives. There is also no need for additional physical equipment because Custom Keys operates on software which use capabilities of all smartphones.

2.2 The Numeric Keypad

Standard QWERTY keyboards, a type of keyboard known to have the letters Q, W, E, R, T, and Y positioned left to right on the top alphabet row, often include a numeric keypad on the right side (Figure 2). Numeric keypads are designed for users to efficiently input numeric keystrokes, similar to how the number keys are positioned on a calculator (*FOLDOC, 5*). However, most lightweight laptops, typically with a 13 inch or smaller screen, do not have a numeric keypad included with the keyboard because the width of the laptop cannot accommodate it. Laptop's manufactured by Apple such as the MacBook, MacBook Air, and MacBook Pro also have no numeric keypad (Apple, 8).

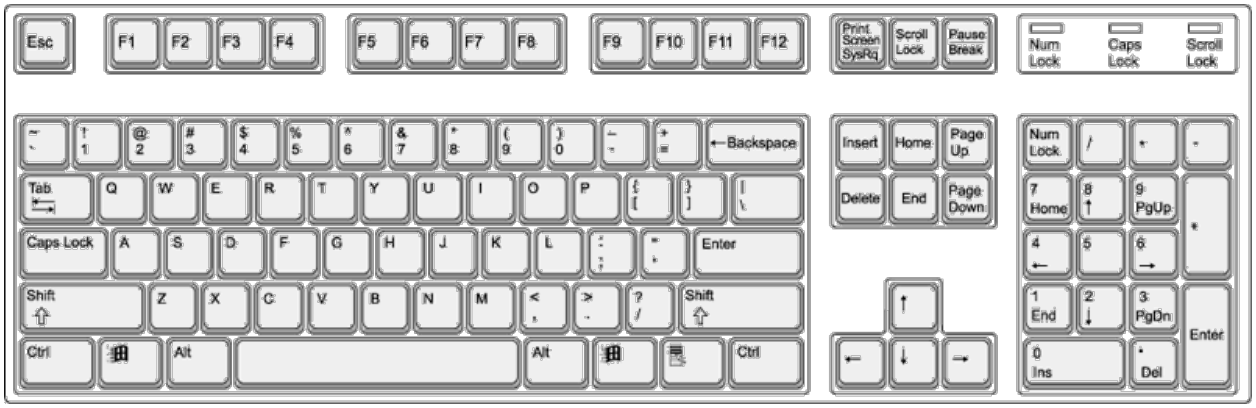

Figure 2. Standard QWERTY Windows keyboard layout with a numeric keypad on the right-hand side (GCCLC, 9).

2.3 The Function Key

Many laptops that run Windows have a function key at the bottom left corner referred to as the “fn” key, located between the control key and the Windows key. Keyboards such as the CMStorm Quickfire, have the “fn” key positioned in the right corner below the shift key (*Coolermaster*, *4)*. The function key is a modifier key which is similar to the ctrl, alt, and shift key. When a user presses a modifier key, it changes how certain keys on the keyboard interact with user input. For an example, when a user pushes the shift key in combination with a letter key, the input produces a capitalized version of the letter. Some laptops such as the the Lenovo’s IdeaPad V460, were designed to have the no dedicated numeric keypad. The Ideapad instead, assigns many numbers and letter keys with an alternative input where one can press a button to change the operation of existing letters and numbers on the keyboard to imitate the design similar to a numeric keypad. The button is the “fn” key and it can be pressed in combination with the letters and numbers to produce the input (*IdeaPad V460*, *7*). However, there is a reason why this design is not sufficient enough to satisfy the users who use the numeric keypad.

The user has to hold down the ”fn” key on the left hand while inputting the other numeric assigned keys. This produces a “similar” feeling of a standard numeric pad for the right hand to input numbers. The number key layout design also tries to preserve the effectiveness of the numeric pad design. However, it restricts the individual's left hand from doing any other task while inputting numbers because the user has to keep holding on to the function key in order to use the numeric pad feature. This is a limitation and most users are discouraged to use it because of this reason.

2.4 Data Communication Implementation

Before I started to search for methods about receiving keystroke signals sent from a smartphone to a computer, I brainstormed possible ways of how data could be transferred. I produced two categories which connectivity between smartphone and computer would be possible. The first category was wired. For the wired category, I thought that USB technology would be the fastest and most reliable implementation. However I had no knowledge of how USB technology works. The second category was wireless connections. I was able to come up with two possible wireless ways for a device to interact with a computer. The first one was Bluetooth. My knowledge was very limited so I needed to do much research on it if I started developing on it. The second is a server and client relationship between the user's computer and phone. And again, much research would be needed before diving into developing the program(s). Although I mentioned two separate categories, I will only talk about the wireless category.

2.4.1 Bluetooth

Bluetooth is a low power and low cost technology which allow devices to communicate wirelessly through a short range. Smartphones typically have a Bluetooth class 2 radio included within their hardware. This means that they support wireless Bluetooth connections up to 33 feet (*Bluetooth, 3*). This Bluetooth technology.

My initial idea implementing bluetooth technology was to program an app so that it would be able to imitate a bluetooth keyboard which can send keystroke data to the computer. After searching for open Android Application Program Interfaces (APIs) and researching for the a lead of what I envisioned my app to be, I came back with no to little leads on how to start or develop an app designed to mimic a bluetooth keyboard. However I didn't stop there.

I bought a bluetooth USB dongle and tested its connection with my Samsung Galaxy S4 smartphone. I paired the devices together and a window prompted me for whether I wanted to transfer files, or play audio from my phone to my laptop. I tested Bluetooth audio streaming. During the Bluetooth audio connection test, I experienced audio cutting in and out every two to three seconds. I was not satisfied with its performance so I decided to move move on with my research.

2.4.2 Server and Client

There are two names which distinguishes connection between computers, server and client. A server a computer which allows client to establish a connection with it, often providing a service for the client and sending responses back to the client. A client is a computer which connects to a server and sends requests to the server. The relationship between a server and client is called Client-Server Interaction (*Hall, 6*). In Custom Keys, the server is the computer and the smartphone is the client. The client would send requests to the server to do a execute a specific keystroke.

2.5 Android Application Development

There were many books and online tutorials to choose to help me develop a foundation in developing Android apps. I chose the Android Developers tutorial. Android's Integrated Development Environment (IDE), Android Studio, was used within the tutorial and therefore it was my IDE of choice. Although there was an alternative option to use Eclipse for the tutorial, I used Android Studio, because it was easier to follow their instructions. The instructions were straight forward and followed Android Studio's IDE interface. I learned many aspects which allowed me to start developing the app such as: Activities, Formatting, and Lists. I explain implementing these technical details in Chapter 3.

Chapter 3: Technical Details

In chapter 3, I explain the technical details of implementing Custom Keys project which includes the method creating a virtual keystroke on the Windows operating system, sending data from smartphone to computer,

3.1 Virtual Keystrokes in the Windows Operating System

Before developing the server and app for Custom Keys, I created a program in C called SendInput.c. When SendInput.c ran, it virtually produced a keystroke without any intervention from the user. It used the windows.h library which had a function also named sendInput which took in three parameters: the number of inputs, the location of the input array, and the size of the input structure. The input structure has information that the sendInput function uses whether it is a keyboard, mouse, or hardware input

each of which have a respective data structure. In this case I stored the virtual-key code of the letter “a” into the keyboard input structure. A list of all the virtual-key codes are listed in a table on Microsoft’s website (Microsoft, 10). After inputting the keycode for the letter “a”, another sendInput function is invoked to release the key, which emulates a single keystroke of the letter “a”.

3.2 Server Development

Since coding for Android was in the language of Java, I decided to program my server in Java as well. I used the java.net library which includes all functions necessary to start a server. The program starts listening on a socket of an available port and prints the current IP address and port number of the server once successful. This feature gives the user sufficient information to connect his or her smartphone to the server. After the server is setup, it runs in an infinite loop listening for incoming connections. The server will close only until the user closes the application.

The server has a switch statement for all the different possible virtual-key codes the client can send. After a connection from the client is established, the server reads the requested keystroke message sent from the client and takes proper action. The server determines the request from the client through the switch statement and then the KeyPress function simulates the requested keystroke. The connection from the client is closed after the request is fulfilled.

3.2.1 Java Native Access (JNA)

Java Native Access, other known as JNA, is a Java API Library that gave me access to the Windows native functions. I used this library to access the SendInput function and INPUT data structures.

3.2.2 Java Swing

Swing is a Java library that allows programmers to create Graphical User Interfaces (GUI). I used Swing to create the GUI for the server to show the current IP address the server is running on and the port number that it is listening on.

3.3 Android Development

When I created Custom Keys, each different screen was a different activity. Activities are the building blocks for the user interface. Each activity has the ability to create a new window. Each activity usually is paired up with Extension Markup Language (XML) code. XML was used to code the user interface for android applications. I also coded the application's process for keystroke requests to the server. The numeric keypad was a crucial part of the design for numeric input so I designed it to be as efficient as possible.

3.3.1 Activities

There were five activities created for Custom Key's: TitleScreen, About, Connect, Configurations, and Numpad. Each activity has a specific purpose. TitleScreen displays a menu of several options to choose from. About explains what the purpose of Custom Keys and gives instructions on how to use the app. Connect displays input boxes where the user can type in the Internet Protocol (IP) address and port number of the server. Configurations shows the a list of key configurations setups. Numpad is currently the only setup. Numpad is the activity that projects a numpad to the screen that the user can use for numeric input.

3.3.2 Connecting and Sending Messages to the Server

The client sends specific messages to the server for each button on the Numpad Activity. I created two functions: getMessage and sendMessage. getMessage is a button listener event that responds when the user clicks a button. The function also determines the exact button pressed by getting information of the tag from XML code and converting it to a string type. After determining which button was pressed, getMessage executes the sendMessage function. The sendMessage tries to establish a connection with the server with the IP address and port number that the user inputted in the Connect activity. If the connection was successful, the message string variable from getMessage, will send over to the sever. If a connection cannot be found, the message will not send.

3.3.3 Numeric Keypad Design

The standard numeric keypad design is a design that individuals would recognize when looking at Custom Keys' numeric keypad. Each of the numbers are ordered in the same positions just like a regular numeric keypad (*Figure 2*). However, I did not include several keys because of room limitations on the screen of the phone. The keys that I excluded were: Num Lock, asterisk, forward slash, and decimal point. Sacrificing these keys were necessary for implementing an optimal button sizes for numeric keypad. Custom Keys' approach to using its numeric keypad has a slightly different approach than a physical one. That is because smartphone screens are touch sensitive, meaning that the slightest touch on the screen will trigger a click event on the phone's screen. On a physical numeric keypad, a user can rest their fingers on the keys lightly without having to worry about mistyping input.

Chapter 4: Evaluation

Chapter 4 has a collection of my evaluations on Custom Keys which includes: Application Performance, Usability, and Drawbacks.

4.1 Application Performance

The connection between the server and app had no problems. All tests were conducted between a Samsung Galaxy S4 smartphone as the client and a Zenbook UX305 laptop as the server. Custom Keys was tested in three locations, my apartment, Starbucks, and on the campus of California Polytechnic State University, San Luis Obispo (Cal Poly SLO). When a user inputs several keystroke requests, the server receives and completes each task within 15 milliseconds from when the user touched the smartphone. Each activity has a load time of one second or less.

4.2 Usability

Depending on the smartphone screen size, users may have difficulty pressing the intended buttons because of image scaling. However there are a many benefits for using Custom Keys. Custom Keys saves space for those who have laptops with no numeric keypad, because when they don't need, they don't need to use it. Whereas laptops with numeric keypads will take up space whether or not it's needed.

4.3 Drawbacks

There are a three drawbacks to Custom Keys. The first, one is that both the server and client must have internet access in order to function. This limits the number of locations the user can use Custom Keys. Another drawback is that it is not for everyone. People who have numeric keypads on their keyboards already will not need to use Custom Keys. And lastly, Custom Keys uses battery life of smartphones.

Chapter 5: Conclusion

Overall, Custom Keys was a successful project. I was able to implement server client interaction, virtual keystrokes, and design my first android app. Although it was a success, there are a couple features that I wanted to add.

5.1 Future Development Plans

If Custom Keys were to be further developed, I would improve it in two ways. I originally developed Custom Keys for the Android platform, but I also wanted a working iOS version. As I mentioned in 2.2, current Apple laptops do not have a numeric keypad. A large portion of the population in the U.S. owns Apple laptops. Therefore an iOS implementation of Custom Keys would make the app even more accessible.

I also wanted to add a button editing feature where the user can create savable and loadable key button profiles. Within the profiles, the user will have the ability to create several buttons. Each button will have an editable shape, size, and function.

5.2 Closing Statements

During this project, I learned how to properly adjust the scope of my project to fit my time and resources. When I initially decided to implement Custom Keys using Bluetooth, I did not realize the depth of research that would be required. Having taken a networking class previously, I was already familiar with the concept of implementing a server-client relationship. The pivot to a server and client based implementation made this project a success.

Bibliography

1. "6 Facts about Americans and Their Smartphones." *Pew Research Center RSS*., 01 Apr. 2015. Web. 10 March 2015. http://www.pewresearch.org/fact-tank/2015/04/01/6-facts-about-americans-and-their-smartphones/
2. "ASUS ZENBOOK UX305 - Features." *Notebooks & Ultrabooks*. Web. 2 March 2015. http://www.asus.com/Notebooks_Ultrabooks/ASUS_ZENBOOK_UX305/
3. "A Look at the Basics of Bluetooth Technology." *Basics | Bluetooth Technology Website*. Web. 27 Feb 2015. http://www.bluetooth.com/Pages/Basics.aspx
4. "Cooler Master Gaming » Products: Quick Fire Rapid-i." *CMStorm RSS News*. Web. 3 March 2015. http://gaming.coolermaster.com/en/products/keyboards/rapid-i/
5. "FOLDOC - Computing Dictionary." *FOLDOC - Computing Dictionary*. Web. 8 March 2015. http://foldoc.org/Numeric%20keypad
6. Hall, Brian. "Client-Server Background." Client-Server Background. Web. 12 March 2015. http://beej.us/guide/bgnet/output/html/multipage/clientserver.html#simpleserver
7. "IdeaPad V460." *IdeaPad V460*. Web. 2 March 2015. http://shop.lenovo.com/us/en/laptops/lenovo/v-series/v460/
8. "MacBook Light. Years Ahead." *Apple*. Web. 10 March 2015. https://www.apple.com/macbook/
9. "Standard Windows Keyboard Layout - Windows Natural Keyboard."*Picture of Windows Keyboard*. Web. 15 March 2015. http://www.gcclc.org/StudentFiles/Mouse-Keyboard/Keyboard.htm
10. "Virtual-Key Codes." *(Windows)*. Web. 20 March 2015. https://msdn.microsoft.com/en-us/library/windows/desktop/dd375731%28v=vs.85%29.aspx