Penti Chorded Keyboard and How to Remap

Documentation for Fall 2017 and Winter 2018

EasyType

Contents

Section I.
Introduction

A. What is the Need for One-Handed Keyboard Development?

Our challenge was presented to us by our need expert who wanted a better way to type and play online games using one hand and minimal finger force. Going into middle school, our need expert wants to be more involved in the digital playground with his peers. A one-handed keyboard would allow him to explore more games that require keyboard input, which currently is limited to small selection of more familiar keyboards such as the standard QWERTY keyboard layout. Incorporating this one-handed, touch screen keyboard will not only give our need expert more access to playing games with his friends, but also help expand his typing skills.

B. Origination of the Penti Keyboard

There are a few open source one-handed keyboards that currently exist. The one that was most easily available for download and with more potential for editing was the Penti Chorded Keyboard developed by Alexander Burger. Information on his development is available on his website:

https://software-lab.de/penti.html

All the code we used for this project is credited to Alexander Burger and Software Labs.

C. How to Use the Keyboard

Detailed instructions for Penti Keyboard usage and the key for chords and its associated output is found on the website previously mentioned above. The following images also are additional visual aid describing how to use the keyboard. The key only shows the chords for letters A-Z and the space key, however, it should be noted that there are many other chord combinations available on the website for all other keyboard functions.

Chord	SHIFT
# ----	SPACE
# --#-	A
# -##-	B
- #-#-	C
# ---#	D
- -#--	E
# #---	F
- --##	G
# #--#	H
- --#-	I
- #--#	J
- ####	K
- ##--	L
- -###	M
- ---#	N
- -##-	O
# ###-	P
# #-#-	Q
# -#--	R
- #---	S
# -###	T
- ###-	U
- #-##	V
# ####	W
# #-##	X
# --##	Y
# ##--	Z
- -#-#	RESET
- ##-#	RESET
# -#-#	RESET
# ##-#	RESET

Figure 1. Key for Chords and Associated Letter

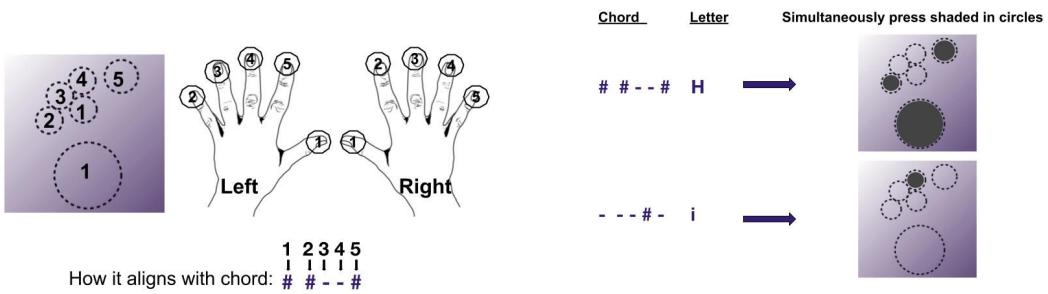


Figure 2. Visual Representation of How to Use Chords

Section II.
Downloading Software and Tools

A. Penti Source Code

The Source Code is also available for download on the Penti Keyboard website at the very bottom of the page:

https://software-lab.de/penti.html

They also provide the APK file for download, which can be downloaded on an Android device and opened through Language Settings (See Section VI. B.). This allows any android device that supports multi-touch input to use this keyboard as a replacement for any default keyboard.

B. Android Studio

To allow our custom version of Penti Keyboard to download, we need to create an Android Package file (APK file). In order to do this, we need to download Android Studio, which is the official development environment for Android systems. This is available for free download, and can be found through this link:

https://developer.android.com/studio/install.html

C. Other Helpful Tools

To view the downloaded source code from the Penti website, it is helpful to have a source code editor or another environment where code can be opened and looked at with ease. A few tools that are free for download include:

Atom: https://atom.io/

Sublime: https://www.sublimetext.com/3

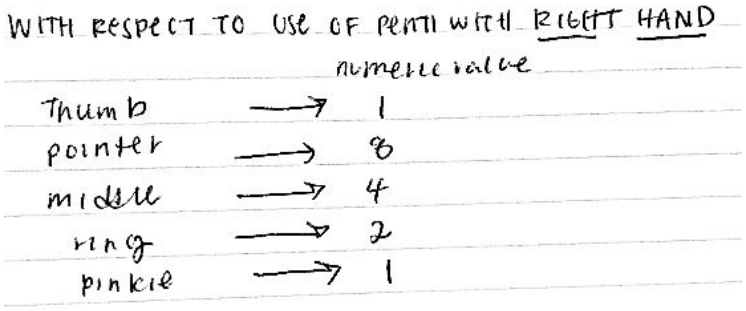
Section III.
Understanding Penti Code

A. Hexadecimals

Looking through the PentiView.java code, we found that each letter has an association with a number. These numbers are called hexadecimals:

https://en.wikipedia.org/wiki/Hexadecimal

The notes for the process of our team figuring out the hexadecimals related to each letter is attached below:



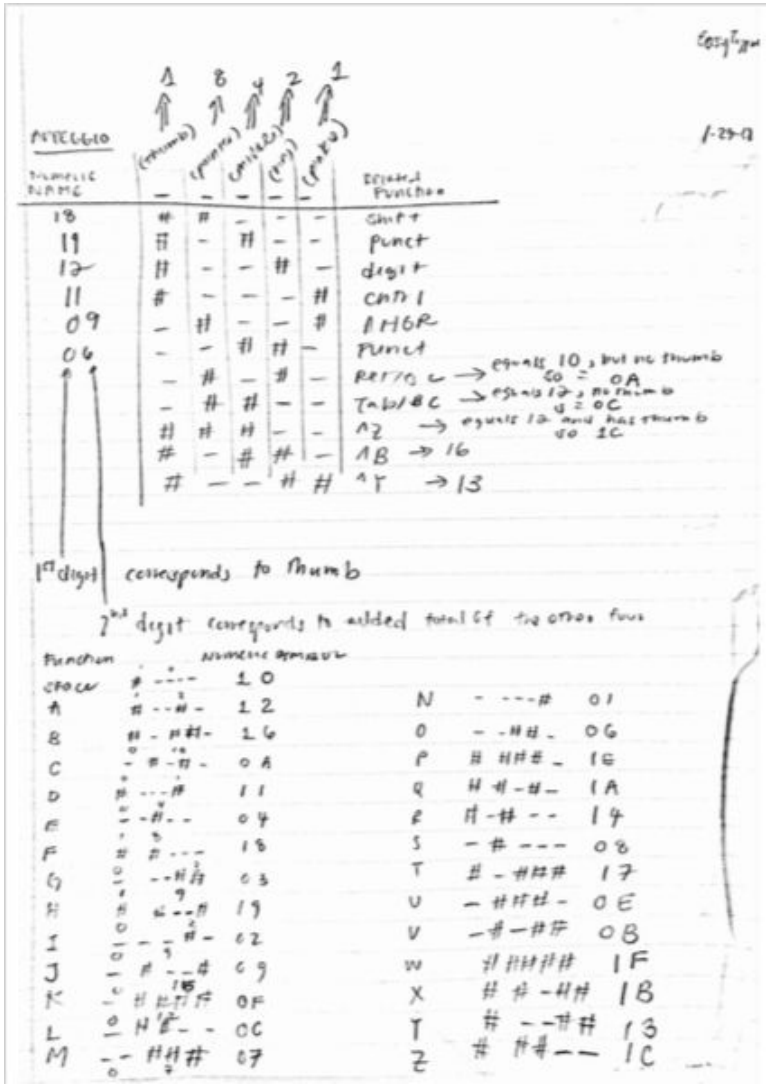

Keyboard
1-29-91
Numeric Name
Related Function
shift
punct
digit
CNTRL
ALT GR
punct
RETURN → equals 10, but no thumb so = 0A
TAB/BC → equals 12, no thumb so = 0C
^Z → equals 12 and has thumb so 1C
^B → 16
^T → 13
1st digit corresponds to thumb
2nd digit corresponds to added total of the other four
Function
Numeric equivalent
SPACE 10
A 12
B 16
C 0A
D 11
E 04
F 18
G 03
H 19
I 02
J 09
K 0F
L 0C
M 07
N 01
O 06
P 1E
Q 1A
R 14
S 08
T 17
U 0E
V 0B
W 1F
X 1B
Y 13
Z 1C

Figure 3. Notes For Figuring Out Hexadecimals

With this we see in numerical order, the letter order corresponding is ‘n', 'i', 'g', 'e', 'o', 'm','s','j', 'c', 'v', 'l', 'u', 'k', 'd', 'a', 'y', 'r', 'b', 't','f', 'h', 'q', 'x', 'z', 'p', 'w.'

B. Manipulating the Array

In the PentiView code, there is an array in that exact same order we just found. Changing the order of the letters listed in this array accordingly change the hexadecimal assigned. Therefore this easily changes the chording of the letter.

```
final static int Penti[] = new int[] {
    0, 'n', 'i', 'g', 'e', 0, 'o', 'm',
    's', 'j', 'c', 'v', 'l', 0, 'u', 'k',
    32, 'd', 'a', 'y', 'r', 0, 'b', 't',
    'f', 'h', 'q', 'x', 'z', 0, 'p', 'w'
};
```

Figure 4. Original Array in PentiView.java

```
final static int Penti[] = new int[] {
        0, 'i', 'n', 'g', 'e', 0, 'o', 'm',
        's', 'j', 'c', 'v', 'l', 0, 'u', 'k',
        32, 'd', 'a', 'y', 'r', 0, 'b', 't',
        'f', 'h', 'q', 'x', 'z', 0, 'p', 'w'
};
```

Figure 5. Changed Array in Custom Version of PentiView.java

Figure 5 shows that letters ‘i’ and ‘n’ have been interchanged. This successfully changes the chord combination that was originally for letter ‘n’ to letter ‘i’ and vice versa.

Changing the PentiHelp String variable part of the code in PentiView.java to align with any changes to the above array may also be important as to avoid later confusion.

```
final static String PentiHelp[][] = new String[][] {
    {"Chord", "S", "P", "D", "A", "F", null, "Arpeggio"},
    {"# ———", "SPACE", "SPACE", "SPACE", "SPACE", "NEW", null, "# #———", "SHIFT"},
    {"# —#—", "A", "`", "6", "8", "F6", null, "# -#——", "PUNCT"},
    {"# -##-", "B", "{", "LEFT", null, null, null, "# ——#-", "DIGIT"},
    {"- #-#-", "C", "]", ",", "ê", "COPY", null, "# ———#", "CNTRL"},
    {"# ———#", "D", "/", ".", null, null, null, "- #——#", "ALTGR"},
    {"- -#——", "E", "[", "2", "é", "F2", null, "- -##-", "FUNCT"},
    {"# #———", "F", "?", "4", "'", "F4"},
    {"- ——##", "G", "=", "9", "à", "F9", null, "- #-#-", "RET/ESC"},
    {"# #—#", "H", "#", "HOME", "ñ", "HELP", null, "- ##——", "TAB/BS"},
    {"- —#—", "I", "!", "3", "-", "F3"},
    {"- #——#", "J", ";", ":", "ş", null, null, "# ##——", "^Z"},
    {"- ####", "K", "@", "DEL", null, null, null, "# -##-", "^B"},
    {"- ##——", "L", "_", "7", null, "F7", null, "# —##", "^^Y"},
    {"- -###", "M", "~", "-", "-", "F12"},
    {"- ———#", "N", ")", "RIGHT", "ñ", "NUM"},
    {"- -##-", "O", "|", "8", "ö", "F8"},
    {"# ###-", "P", "}", "+", null, "PASTE"},
    {"# #-#-", "Q", "'", "PGUP", null, "QUIT"},
    {"# -#——", "R", "$", "5", "€", "F5"},
    {"- #———", "S", "*", "1", "ß", "F1"},
    {"# -###", "T", "%", "^~"},
    {"- ###-", "U", "&", "0", "ü", "F10"},
    {"- #-##", "V", "(", "DOWN", "ü"},
    {"# ####", "W", "<", "INS"},
    {"# #-##", "X", "\\", "PGDOWN"},
    {"# —-##", "Y", "^", "UP", "ÿ", ""},
    {"# ##——", "Z", "\"", "END", "ş", "F11"}
};
```

Figure 6. Penti Help Code in PentiView.java

C. Other Possibilities

Further development of the code can be include looking into changing arpeggios.

Section IV.
Android Studio Setup

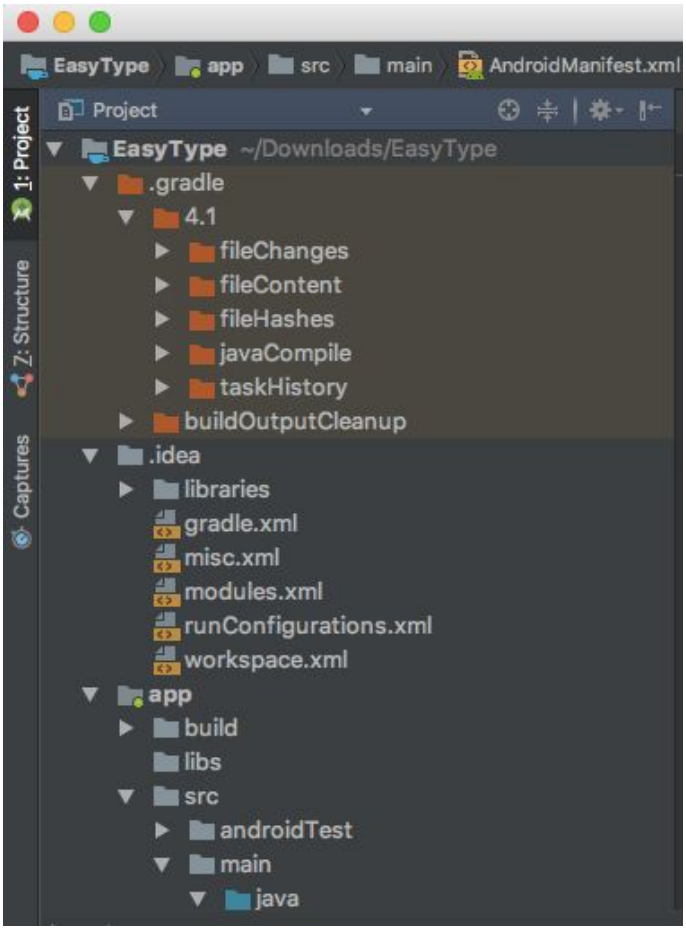
A. Emulator Setup

Setting up and emulator through Android Studio is not necessary if you have the Android device connected to Android Studio while editing the code, but it can be helpful to have to quickly check during the process if the app opens correctly.

1. Go to Tools, Android, AVD Manager, Create Virtual Device
2. We used a Tablet, Nexus 7, Nougat API level 24 Android 7.0

B. Settings for Building a New Project

1. Open Android Studio and create new project. Our version had it built under this order:



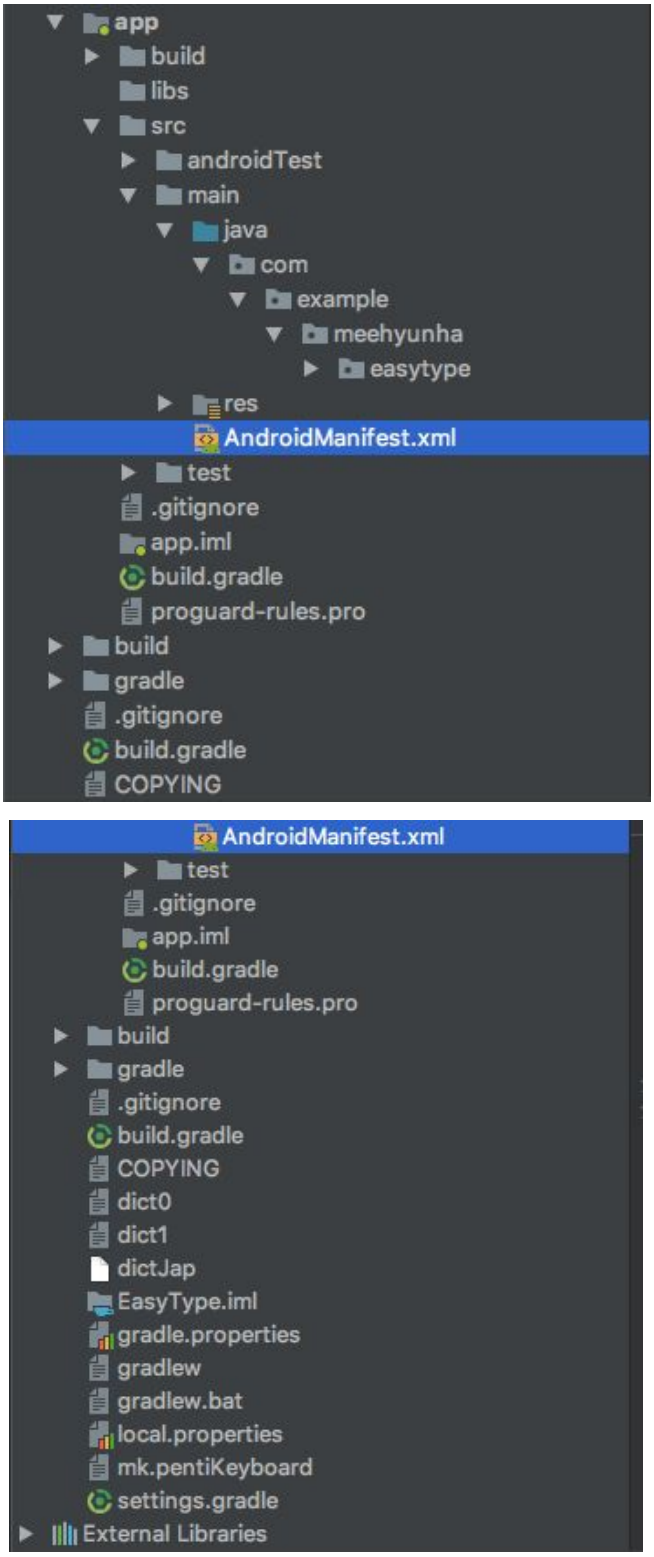


Figure 7. Left Bar on Android Studio Should Display Files in Such Order

2. **This specific path of our custom version will now be referenced throughout the steps for ease of following along**. Note that your path name will be different if you reference Section V. A. However, it will be helpful to name all other files exactly as the source code labels them.
3. Configurations are set to app and setting the launch to nothing is important:

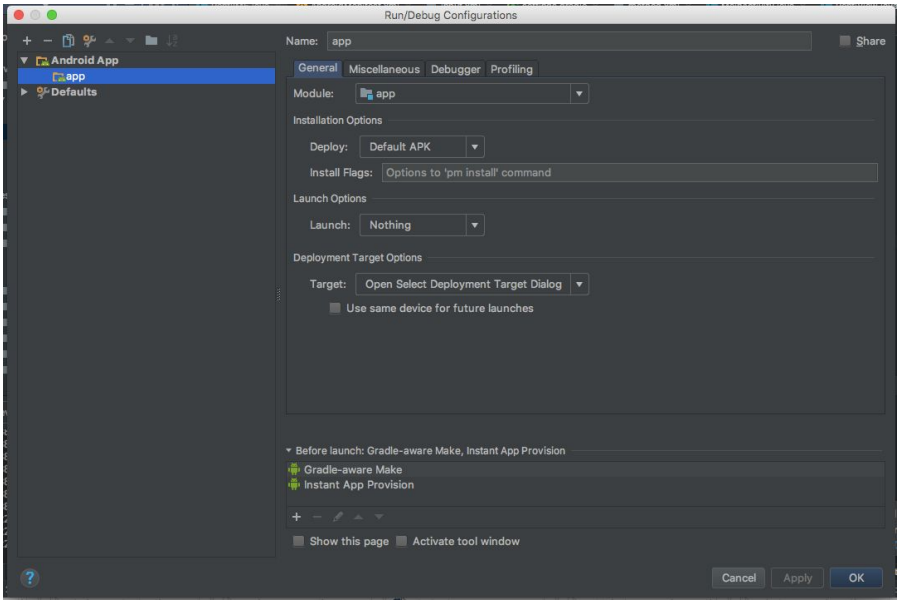


Figure 8. What Settings Should Look Like Under Configurations

Section V.
Building Gradle on Android Studio

A. File Location and Path

When starting a new project, noting the exact location of the new project is important as this will have to be referenced many times within the code. If the location of files is not correctly named, the code will not know where to find certain classes or functions that are being called.

An example of this would be if writing a code where an image file needs to be analyzed, somewhere in the code there needs to be a script that directs it to the location of the image file. The following link has more detailed information on paths:
https://docs.oracle.com/javase/tutorial/essential/io/path.html

For this project, because we created a new project, our path name is different from the one referenced in the original source code we downloaded from the Penti website. Due to this, it is important when copying and pasting the code from the original source code, to change the path name. In this Java environment, this is seen as 'package com.example..etc' format on the top of each Java Class and within other files in a 'package = com.example..etc' format. The images below show the difference of labeling the package in the original PentiIME.java and our version.

```
1  // 16may16abu
2  // (c) Software Lab. Alexander Burger
3
4  package de.software_lab.pentikeyboard;
5
6  import android.inputmethodservice.InputMethodService;
7  import android.view.View;
8
9  public class PentiIME extends InputMethodService {
10    PentiView PV;
11
```

Figure 9. Original PentiIME.java Class by Software Lab

```
PentiIME
package com.example.meehyunha.easytype;

import android.inputmethodservice.InputMethodService;
import android.view.View;

public class PentiIME extends InputMethodService {
    PentiView PV;
```

Figure 10. EasyType's PentiIME.java Class

B. Coding

The Source Code for the Penti Keyboard is all written in Java. Some good resources for understanding Java language:
https://docs.oracle.com/javase/7/docs/api/allclasses-noframe.html
https://docs.oracle.com/javase/tutorial/java/index.html

C. Copying and Adding Necessary Files

I found that it helps to open two finder windows and have the original Software Lab downloaded version of Penti Source Code open in one window and the custom Android Studio version of the Penti Source Code that we are building open in the other. Comparing the contents of each file in both versions allows us to catch what parts are missing and different in the new version.

Steps for a successful initial run:

1. Go under app file and under 'easytype' and create two new Java classes. Copy the PentIME.java code from the original source code in one of the new created Java class, and PentiVIew.java code in the other. Name them 'PentiIME' and PentiView,' respectfully. Additionally, add another Java class under 'easytype' folder called MainActivity.

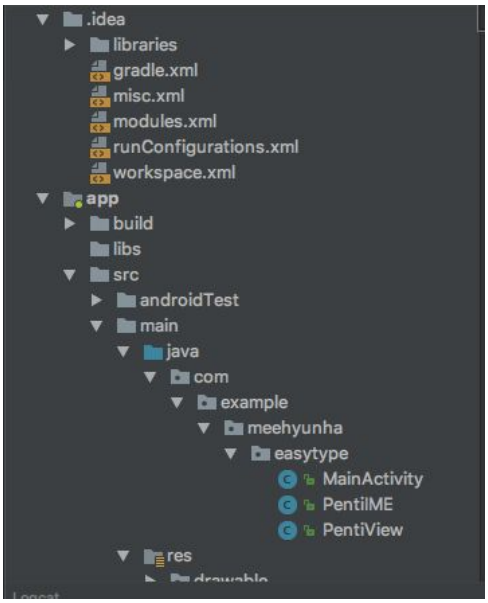


Figure 11. Java Classes Needed

2. MainActivity.java will be mostly empty, only consisting of the following code:

```
package com.example.meehyunha.easytype;

import android.app.Activity;

/**
 * Created by meehyunha on 2/14/18.
 */

public class MainActivity extends Activity {
}
```

Figure 12. Main Activity

3. Copy image files.
    a. Under ‘app,’ ‘src,’ then ‘res,’ make sure all files look like this:

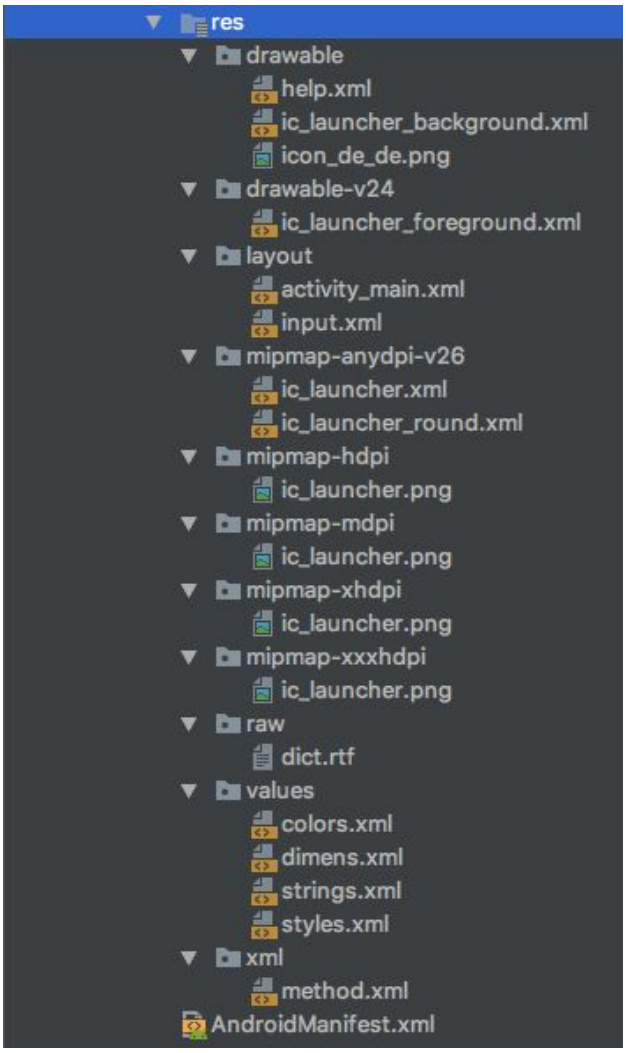


Figure 13. Res Folder

    b. Delete any .png files that are not icon_de_de.png or ic-launcher.png. Do not delete any other things.

c. Under the ‘raw’ folder, the dict.rtf is the same as the file labeled ‘dict’ under the original source code. I ran into the problem of just copying the file over to my raw folder it to not work. Open the original ‘dict’ file using a text editor application, then copy paste that and save it as your own ‘dict’ file in the text editor. Then transfer it over to your ‘raw’ folder.

d. The method.xml should look like this:

```
<?xml version="1.0" encoding="utf-8"?>

<input-method xmlns:android="http://schemas.android.com/apk/res/android"


>

    <subtype
        android:label="@string/label_subtype_de"
        android:icon="@drawable/icon_de_de"
        android:imeSubtypeLocale="de_DE"
        android:imeSubtypeMode="keyboard" />
</input-method>
```

Figure 14. Method.xml

4. Run and build gradle!

D. Android Manifest

One important resource to read through before editing some of the generated code files is the following link, which describes creating an input method:

https://developer.android.com/guide/topics/text/creating-input-method.html#IMEAPI

Once the code is initially built, an xml file labeled AndroidManifest.xml will be found under the folder ‘app’, then ‘src’, then ‘main’, then ‘res.’

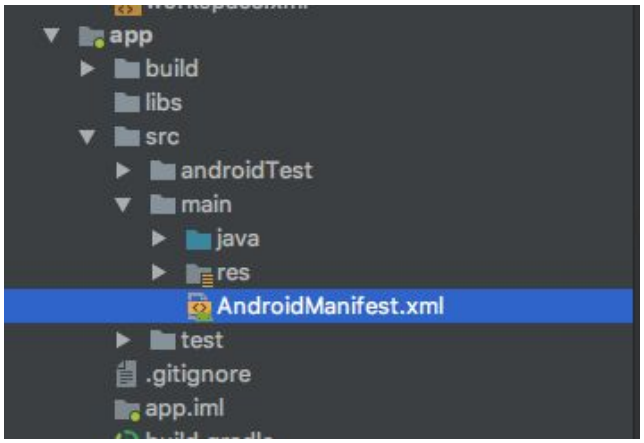


Figure 15. AndroidManifest Location

Open this xml file and make sure it looks like:

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.meehyunha.easytype">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="EasyType"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <service android:name=".PentiIME"
            android:permission="android.permission.BIND_INPUT_METHOD">
            <intent-filter>
                <action android:name="android.view.InputMethod" />
            </intent-filter>
            <meta-data android:name="android.view.im" android:resource="@xml/method" />
        </service>



    </application>
    <uses-permission android:name="android.permission.WRITE_SETTINGS" />

</manifest>
```

Figure 16. AndroidManifest

E. Debugging

Running into any issues during the build, opening logcat and event log on Android Studio is a helpful tool to see what is triggering an error. These two tools are found at the bottom bar of Android Studio.

Section VI.
Setup on Android Device

A. Downloading the Application

After downloading the APK file, locate it on your desktop and send it via email to android device or connect the android device to android studio and choose to run on this personal device instead of on the emulator. After download on the Android device, it should appear under Settings - Applications and also on Languages & Input.

B. Language Settings

Once the Penti Keyboard APK file is successfully downloaded onto the Android device, it can be accessed through the going to ‘Settings’ then ‘Languages & Input’, and ‘Virtual Keyboard’ under ‘Keyboard and Input Methods.’ The Penti Keyboard should be found.

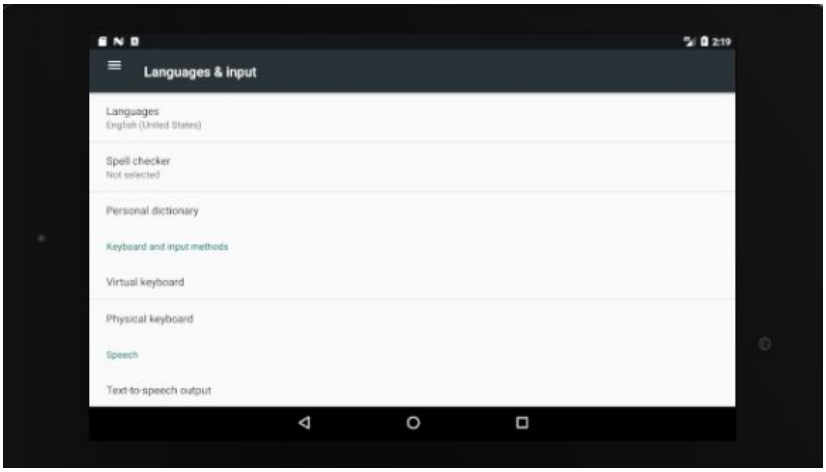


Figure 17. Where to go on Android Device to Access Penti

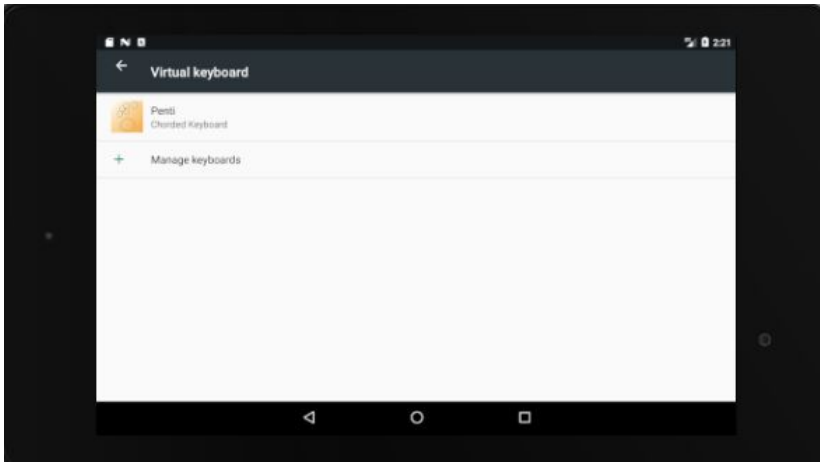


Figure 18. What Penti Should Look Like Under Virtual Keyboards

Once the keyboard is switched on, open a browser and try typing. This should automatically be followed by the letters 'Penti' appearing on the screen, which means it is waiting for the user to press all five fingers down to create the typing locations of the keyboard. This is also better explained in the Penti Website.

C. Using Remote Control Server

For our project, we needed to find a way to incorporate the Penti Keyboard with a Windows based computer. To achieve this, one can use their Android device with the downloaded Penti as a wireless keyboard. The best method we found was downloading a remote control application on both the Android and computer. This application is free and available for download through the following link:

http://remote-control-collection.com/download/server/windows/

Section VII.
Future Goals

A. Remapping Whole Keyboard

For the past two quarters, the keyboard was mapped according to our need expert's primary need for access to any one-handed keyboard for his left hand. This meant that there was little remapping required due to his ability to easily use all 5 fingers. In the future, the next goal would be to remap the entire keyboard according to his right hand mobility. The process would look something like this:

1. Gain more information on need expert's right hand and which fingers are the strongest
2. Research letters and key functions that are most used (If possible for his age group)
3. Change chords according to this research and his preferred amount of fingers incorporated
4. Test newly chorded keyboard with need expert and gather data on speed and ease of typing to continue to improve the keyboard

B. Adding Trackpad Function

A more challenging task for the customized Penti keyboard would be adding another input to the keyboard that functions as a mouse. This would be more efficient and require less setting up for our need expert when typing or playing online games.