International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 08 | Aug 2020 www.irjet.net p-ISSN: 2395-0072

Phoenix Gaming Assistant App - an Assistant App for the Physically Disabled

Anjali E. Chaudhari[1], Sharvai Patil[2], Shivam Mahajan[3], Siddharth Umachandar[4], Prof. Rohini Nair[5]

[1-4]Student, K. J. Somaiya College of Engineering, Maharashtra, India
[5]Professor, K. J. Somaiya College of Engineering, Maharashtra, India

---

Abstract - *In this paper, we shall be laying out the process for the development of an Android application that uses Android accessibility events and Speech-to-Text technology to allow users to play Android games using only their voice. For the Speech-to-Text technology we shall be using the Porcupine framework developed by Picovoice. The Android accessibility services are used to execute touch events on the device's screen at a location called the 'Pressure Point'. The user chooses where to place the pressure point and thus, where to carry out the touch event when the trigger word 'porcupine' is said by the user. We also analysed the time between the utterance of 'porcupine' and the execution of the touch event for various Android devices.*

Key Words: **Android Development, Android Accessibility Services, Porcupine, Pressure Point, Speech-to-Text**

1. INTRODUCTION

With the advent of mobile technology, most desktop applications have been adapted to be used on mobile devices, especially smartphones. Games are a big chunk of these applications. From idle tap-tap simulators to fast paced first person shooters, games have become a part and parcel of the "Mobile Experience."

Unfortunately, today, people that are missing a limb or both, are devoid of the enjoyable experience of playing mobile games. There are hardly any games on the app store that are made to accommodate the needs of these people. Besides, games that are speech driven also exist in miniature numbers. Plus, there is no application that enables physically handicapped persons to play any game using their voice.

Our project aims at allowing users to play android games using only their speech. We will be using speech-to-text technology to achieve this task. First, the user will place a 'pressure point' that is, a point on the screen whose coordinates are saved by the app. When the player speaks the word 'porcupine', a touch event will be executed at the pressure points location thus, making the game deal as if someone is playing with their fingers.

Our aim is to help specially-abled players, get an equally immersive experience as that of normal players. Also, we plan to release this app on the Google Playstore in the coming future to allow anyone with an android device to play.

2. LITERATURE SURVEY

Reference [1] makes a comparative analysis of the feature extraction techniques LPC and MFCC on the basis of noise analysis. LPC shows more deviation for similar kinds of noise than MFCC does.

Reference [2] essentially lays down an overview of the entire speech to text process. It identifies the steps that need to be carried out for speech recognition.

Reference [3] gives an in-depth view of the MFCC Feature Extraction technique that we are going to use for our project.

3. OBJECTIVES OF THE STUDY

Our objective is to propose a model for an Android application that shall use the porcupine framework provided by Picovoice and Android accessibility services that shall be used to execute touch events on other android applications efficiently.

4. PROPOSED SYSTEM

4.1 System Architecture

We are proposing an app that shall behave as an overlay over the games that the user wishes to play. Our app can then be used to control these games using the Speech-To-Text technology. The app will allow players to set a 'Pressure Point' on any part of their device's screen which can then be controlled with their speech, allowing the players to play most single-tap Android games available today (eg. Tap-Tap Dash, Stack, etc.)

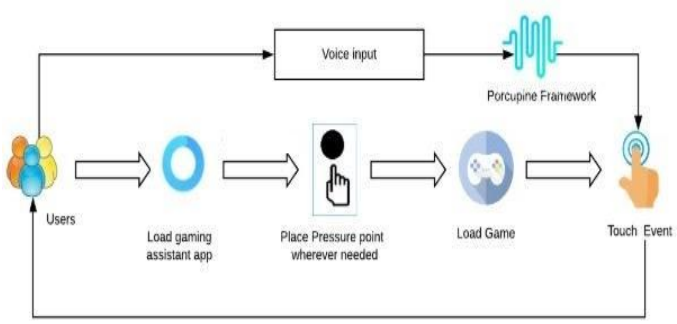

Fig -1: System Architecture

- **Pressure Point:** The pressure point is a coordinate which is mapped to a label 'porcupine'. When the label is said by the player, a touch event shall be executed at the spot of the coordinate.
- **Porcupine Framework:** This is a Speech-To-Text framework for android developed by the company PicoVoice. The framework constantly listens to the voice input from the mobile device's microphone. These voice samples from the user are analyzed and using speech to text recognition, and it is determined if the user said the word 'Porcupine' or not.
- **Touch Event:** Once the user says the word porcupine, a touch-event is executed at the location given by the coordinates on the screen thus allowing the user to play the game using only their voice.

5. IMPLEMENTATION DETAILS

5.1 Placing of Pressure Point on the Screen

Before the app can begin executing touch events, a pressure point needs to be placed anywhere on the display. This is done by loading a screen where the user needs to long press at the location where he wishes to have a pressure point. The steps involved in this process are:

- The user has to long press to place a pressure point. This long press is detected using the onLongPress() function which is provided under the GestureDetector class.
- Using getX() and getY() functions, we get the x and y coordinates of the point where the user does the long press.
- The coordinates that we fetched are sent to the PorcupineService class using an Intent using the function putExtra().
- In the PorcupineService class, using the intent, we get the coordinates that we sent from the MainActivity class using getIntentExtra() function.

The packages used for this to be carried out are:

- GestureDetector: this is used to detect the long press
- Intent: this is used to transfer data to the PorcupineService class

The functions used are:

- getX()
- getY()
- putExtra()
- getIntentExtra()

5.2 Speech-to-Text Recognition

The speech to text recognition ability is provided by the Porcupine framework for android. We provide a path to the model and the keywords along with the sensitivity to the framework. It fetches audio input from the device microphone and analyses it in real-time. The steps involved in this process are:

- We use the PorcupineService class for Speech-to-text recognition.
- We create an object of the PorcupineManager class inside the PorcupineService class.
- We pass the modelFilePath, keywordFilePath and the sensitivity to the PorcupineManager constructor. The modelFilePath is the path to the speech-to-text recognition neural network model and the keywordFilePath is the path to the trigger word file (.ppn) which is included with the porcupine framework.
- We then call the star() function in the PorcupineManager class to begin the speech-to-text recognition.
- To send a notification about the number of times the keyword was recognized successfully, we use the NotificationManager class and create its object and pass the contentTitle, contentText, appIcon and the contentIntent to its constructor and then call its build function.

The packages used for this to be implemented are:

- PorcupineManager: this is used for providing the speech-to-text detection
- PorcupineManagerException: this is used to handle any exceptions that might arise during the speech-to-text detection
- NotificationManager: this is used to provide the ability of sending notifications through the app.

The functions used are:

- start(): to start the PorcupineService
- build(): this is used to build a notification

5.3 Touch Event Execution

Touch events are executed using Accessibility services. These need to be enabled for the app from the device settings menu. The steps involved to execute touch events are:

- We create a function called createClick() which takes the x and y coordinates of the pressure point and returns a GestureDescription.
- A path to the location where the touch event is to be executed is found by creating an object called clickPath of the class Path. We then call the function moveTo() and pass the x and y coordinates of the Pressure Point.
- Then, using the clickstroke and clickbuilder objects of GestureDescription.StrokeDescription and GestureDescription.Builder classes, we execute the touch event at location specified by x and y.
- When executing the touch event, the onCompleted() function is called and overrides its base class function.
- Once the touch event has been executed, the onCancelled() function is called and overrides its base class function.

The packages involved in the execution of touch events are:

- AccessibilityService
- GestureDescription
- Path

The functions involved in this are:

- dispatchGesture()
- createClick()
- moveTo()
- addStroke()
- build()
- onCompleted()
- onCancelled()

6. RESULTS AND DISCUSSION

6.1 User Interface Screens

In the following screenshots we have displayed the implementation for our proposed application for two games: Tap-Tap Dash and Stack.

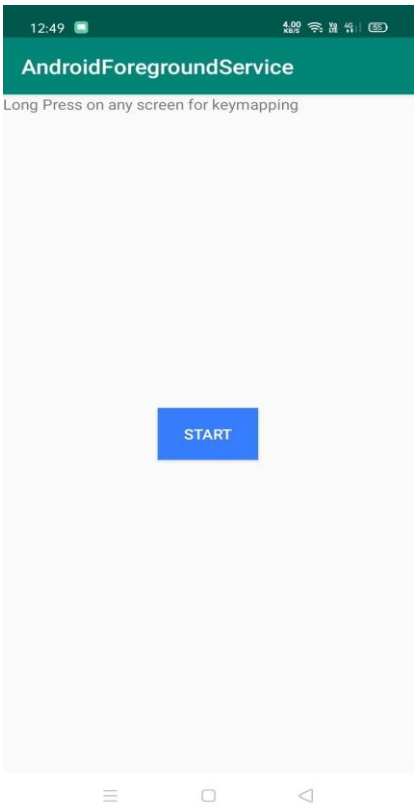

Fig -2: Opening the Gaming Assistant App

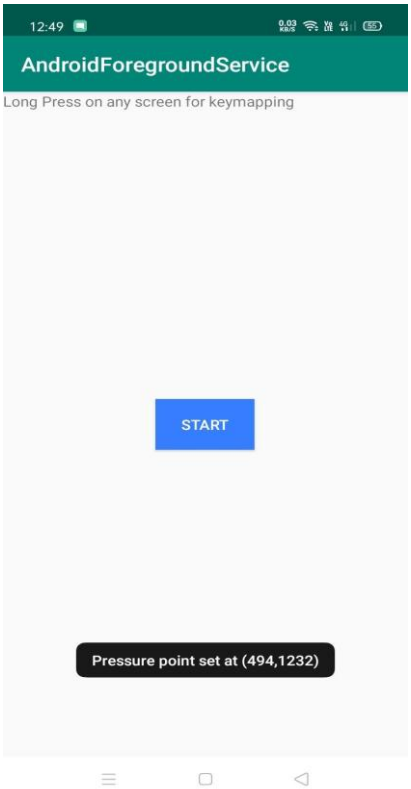

Fig -3: Placing the Pressure Point at (494, 1232)

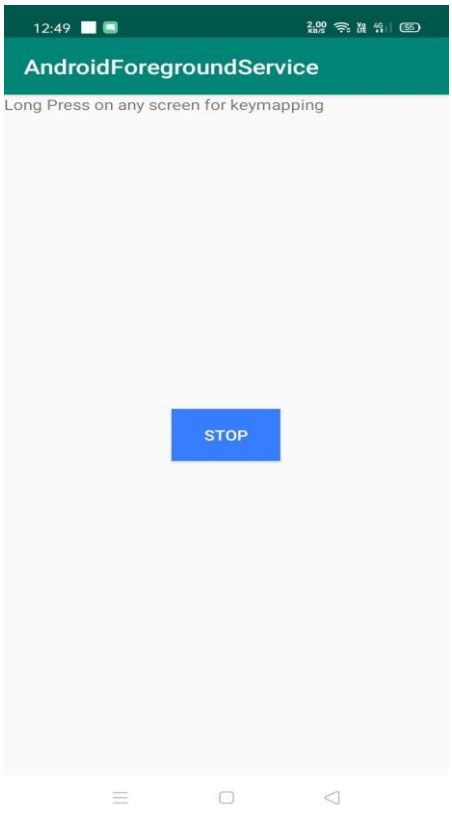

Fig -4: Start the app

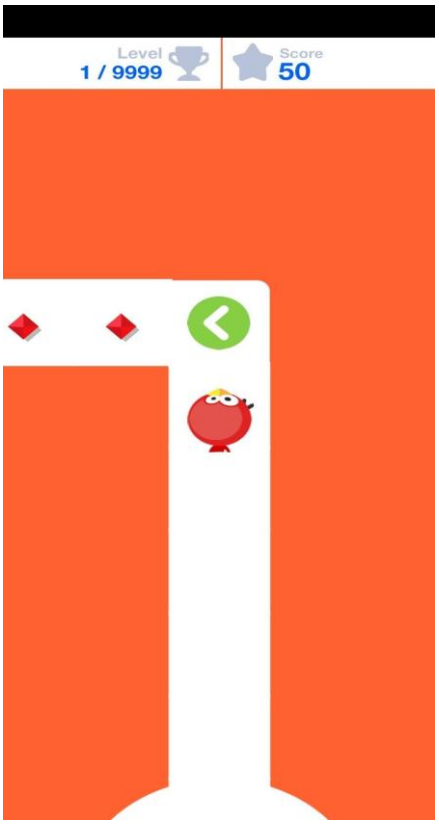

Fig -6: Before the Touch Event

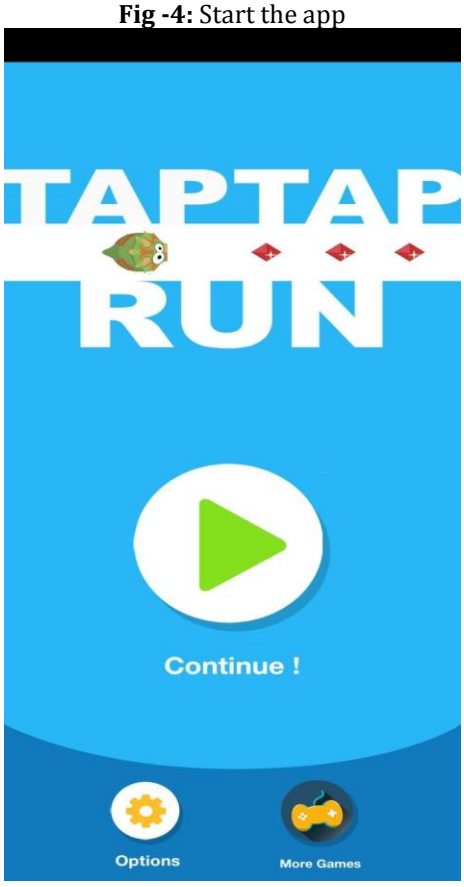

Fig -5: Open the Game Tap-Tap Dash!

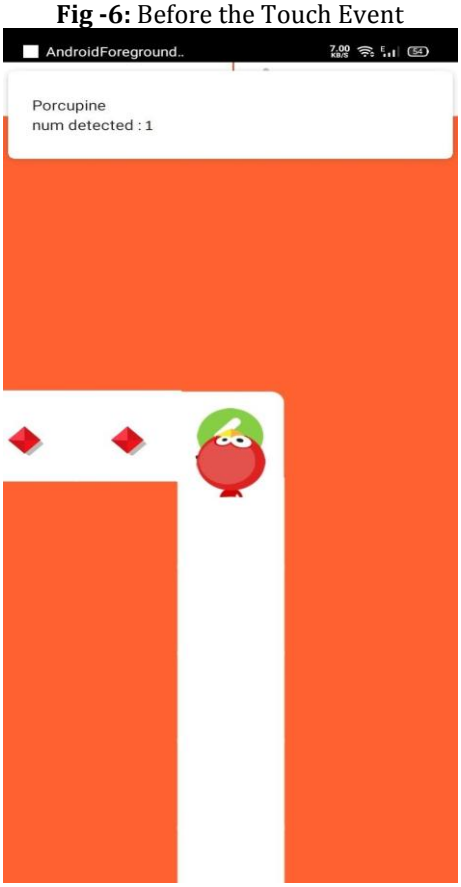

Fig -7: "Porcupine" detected

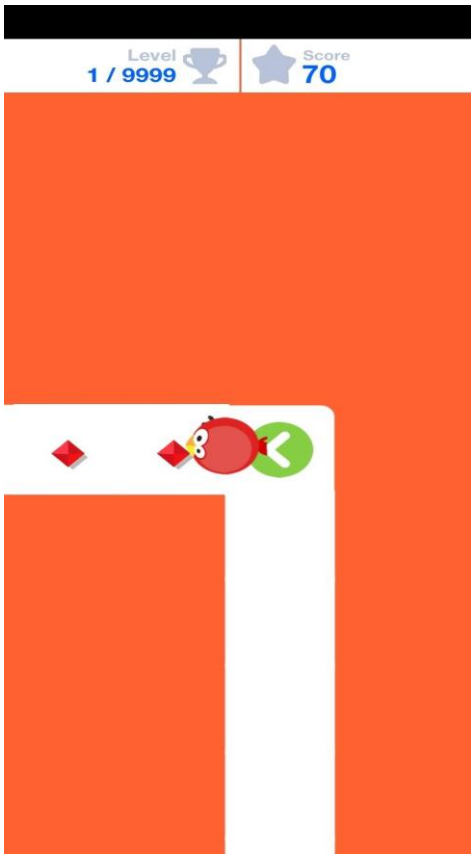

Fig -8: Touch Event executed

Fig -9: Testing the game "Stack"

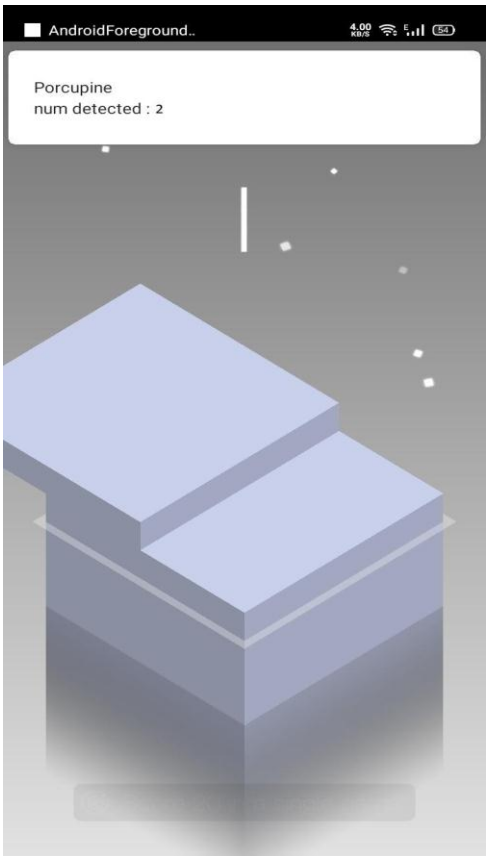

Fig -10: "Porcupine" detected and block placed

6.2 Response Time Analysis

The helper can load the overlay app and can place the overlay anywhere on the screen. There is an "X" icon on top of the overlay that can be used to close this application.

The following are the observed response times that were calculated using an external hand-held timer.

Table -1: Response Time for the game 'Tap Tap Dash!'

Sr. No.	Mobile Device	RAM	Processing Speed	Response Time
1	OnePlus 3T	6GB	1.6 GHz	< 1sec
2	Redmi Note 4	4GB	2 GHz	< 1sec
3	Redmi 7A	2GB	2 GHz	< 1sec
4	RealMe C3	3GB	2 GHz	< 1sec

Table -2: Response Time for the game 'Stack'

Sr. No.	Mobile Device	RAM	Processing Speed	Response Time
1	OnePlus 3T	6GB	1.6 GHz	< 1sec
2	Redmi Note 4	4GB	2 GHz	< 1sec
3	Redmi 7A	2GB	2 GHz	< 1sec
4	RealMe C3	3GB	2 GHz	< 1sec

Since there is no way to calculate the response time internally without increasing the complexity of the code such that the response time in-turn gets affected. Therefore, as observed all response times are noted to be less than 1 second.

7. CONCLUSION

We have developed an application that we hope shall help specially abled people to play mobile games that they usually can't enjoy otherwise. In order to achieve this, we have used speech-to-text conversion technology to allow the players to use voice commands in order to execute touch events on their device's screen. Our main aim was to provide small response times so as to guarantee an immersive gameplay. We have selected 2 games in order to analyze and compare the response times on 4 mobile devices and games. The result of this study is that for all the tested devices and games, the app gives a response time that is less than one second. Thus, we can extrapolate this conclusion to claim with confidence that the app gives a response time in this range for most mobile devices and games.

As per our aim to provide an immersive experience, we are satisfied with our proof of concept and are confident that this shall bring a positive impact in the lives of specially abled people.

8. FUTURE SCOPE

- Including custom trigger words that act as labels for pressure points.
- Ability to add more than one pressure point.
- Adding swipe and long hold events.
- Release to the google play store.

ACKNOWLEDGEMENT

We would like to extend our thanks to K. J. Somaiya College of Engineering, for giving us the opportunity to engage in this project.

We would also like to thank our mentor Prof. Rohini Nair for her constant help and support throughout the project. We could count on her help in rectifying our mistakes and steering us in the right direction every step of the way.

Finally, we would like to thank everyone that assisted us directly as well as indirectly through the process of the development of the project.

REFERENCES

[1] A comparative Performance Analysis of LPC and MFCC for Noise Estimation in Speech Recognition Task by Syed Sibtain Khalid, Safdar Tanweer, Dr. Abdul Mobin, Dr. Afshar Alam (2017)

[2] Speech Feature Extraction Techniques: A Review by Shreya Narang, Ms. Divya Gupta (2015)

[3] An Approach to Extract Feature using MFCC by Parwinder Pal Singh, Pushpa Rani (2014)

[4] Picovoice Porcupine framework: https://github.com/Picovoice/porcupine