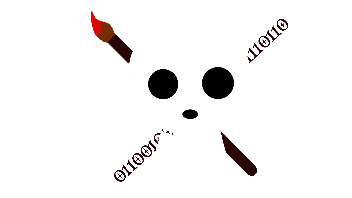
A picture containing light

Description automatically generated

|  |
| --- |
| Voice recognition documentation 2020 |
|  |
| March 3th 2020  Dev Pirates  Made by: Ian Hartog |



Layout

Layout.……………………..…………………………2

Preface..…………………….………………………..3

Choice accountability………...……………………4/5/6

Documentation Voice app…………………………7/8/9

Sources…...…………………………………………..10

# Preface

|  |
| --- |
| This document is meant to inform you, the reader why we chose the speech recognition library we are using for our program, and what road we walked to finish it. Our voice recognition program is called BuggySpeech. The program listens to the user and checks if the input is equal to one of the commands in the commands list. The user can also add custom commands and add button presses to them. What happens when a command is recognized is that it looks at what the command is supposed to do and in this case push a certain button, for example the e button. With this you can control our game made to showcase this, or you could use it for any other game or app like Microsoft word as example. |
| *“The difference between screwing around and science is writing it down” ~ Adam Savage* |
|  |

|  |
| --- |
| Choice accountability  On this page I will let you, the reader know why we chose the voice recognition library we are using now and why we chose to make a separate app for the voice recognition.  Lets start with why we chose a separate app instead of including it into our game. The fun thing is that our original idea was to build the voice recognition into the game. We decided to make a separate app for the next reasons: we found out that Godot did not support Microsoft ddl’s so it would give an error trying to build it in there. Second and probably a more important fact is that with building an app you can use it for more than just playing our game. You could also use word, google or any other app with our BuggySpeech. This made us choose the stand-alone app.  Microsoft speech recognition  Now let’s get to why we chose our library. First off let’s start with which library we chose. The library we decided to use is the Microsoft speech recognition. Or as it’s ddl name: System. Speech. We are using the recognition function in the library to receive commands, and we are also using the Synthesis for fun to get some spoken text back, which we will use in the presentation of our product. The reason why we chose this library is because you can add unlimited custom commands, it runs on the voice recognition build into your computer which makes it very easy to train with multiple voices and I also had a little experience beforehand with this library.  Speechmatics  Here’s the first one we researched but didn’t use: Speechmatics. There weren’t many problems with this one, except the fact that it costs money for every query, which would make it kind of expensive.  IBM Watson  Next up is IBM Watson. Sherlock and Watson would probably be disappointed that we didn’t use this one, but there are some good reasons for that. It needs much time to use it to its full potential, it regularly needs maintenance and it costs money.  Diagflow  Diagflow is one that looked good but ended up not working for us. It doesn’t support that many languages which could be a problem if we do want that. Also the library has more functions then we actually need, which would take up more space while not needed.  Microsoft cognitive services  And now we have Microsoft cognitive services. This one is also made by Microsoft, just like the one we’re using. But there are some problems with this one. The main thing is that it uses cloud processing which we don’t want because it not being able to be used without Wi-Fi. Second problem being that it’s meant to be used for things like special microphones or other devices.  Google API  Google might want our location, but we don’t want the google API. The reasons for that are quite simple. It costs money and it has a limited self-build grammar library. The last one was a big problem for us, since we want the user to be able to make their own commands instead of only listening to specific ones.  Alexa  Last but not least, here’s Alexa. This one was quite the good library, except for a few points. It uses cloud processing like many others, and it’s build for specific things. You’ve probably seen this ad or maybe a product review where these people are talking is their headphones or Bluetooth speaker. That’s what Alexa is build for. To be used on devices that support Alexa and then use commands that allow things like google searches. But that’s just not what we’re after so we left Alexa at home for this project. |

*Documentation voice app*

Now we’re at the point of the documentation itself. Here I’ll show you what happened during the development of this app and what we had to change or adjust. Our idea started off as just a script in our game that would allow voice controls. It would then see if it got a certain text (string) to make a button work. After 3 days of development in Godot and visual studio code we found out that Godot didn’t support Microsoft libraries, we found out late because we didn’t do too much research on the game engine which we should have done.

I was annoyed by the fact that I couldn’t use the library I wanted that I almost decided to completely write my own voice recognition in c#. I was unable to find any information on how to write such a difficult A.I., so I started thinking how we could solve that problem. I started thinking, and something came to my mind. I have an app called voice attack on my laptop which I use for basically the same reasons. I can play my games with voice commands thanks to that app. So after realizing that I had that app I came up with the idea to try and make something very similar to that. So I decided to also make an app that would work in anything.

That was a massive advantage of making an app. Being able to use the app in anything would make it more useful, and also be more of a success if we were ever to publish any of this. Now with Godot and visual studio code gone I could go back to my old and reliable visual studio. I decided to make a windows forms application, mainly because there was a lot to find about that, but also because it would allow me to design an app very easily. I had no experience with a forms application, so development would be interesting. I looked up how it worked and how I could get basic voice recognition working in my program. After that was done a few days had passed.

Now I wanted to simulate button presses with my commands. My idea was to let the switch case run a function based on the command, but due to some argument problems that wouldn’t work. To this day I still have no idea why. I had to think of a different way to do this. I decided that if I could find a way to simulate a key press I could also use that in the switch case itself. First off I thought I had to do this with just input.KeyDown, but that turned out to be completely wrong. I spend a whole week to find out how to fix it, and even had some help from a teacher. Finally I found this thing called SendKeys.SendWait. What this does is it presses a key when something happens (like for example a voice command). That was exactly what I needed. Now I had this, everything should work, right? Nope, it didn’t do anything. I tried everything I thought could prevent the problem, but to no avail. After 5 days I finally found out that I needed to add some brackets () in these strings “” to make it work. I was really annoyed that that took me 5 whole days, but after testing my program, it finally worked! Now I wanted to make the app a bit nicer.

I made some buttons to mute and unmute the recognition, but after screwing around with the click event, my buttons stopped working. 2 days later I finally fixed it somehow, but that same day I actually broke it again….oops.

I decided to look up how I used to have my code before I broke it, but even after that it didn’t work. I thought to myself, I can keep playing with this code and spent too much time on it, but I can also start over. So I started over. It was less work this time since I knew what I had to do and what functions and such I had to use. I had my entire program redone in 3 hours, that’s fast.

After that I replaced the buttons with clickable images, and boom, working program. I thought the program looked a bit empty because I had two buttons in it, so I added a dark and light theme, just for fun. It was super easy and only cost me about 5 minutes. So now I had 4 buttons, two for turning the voice control on and off, and the other two for toggling between light and dark theme. The program still looked empty and could use some more space in the middle. My next idea wat to actually print the command the user was saying, this was also easy and took 20 minutes. After that the program was looking fine and was working properly.

The next day me and my partner Jordan were going to test the program in the game we were making for it. In the beginning it worked, but still with message boxes. These message boxes were used as a debug for me to test if everything worked, but now that I knew that it wasn’t necessary anymore, so I removed them. It stopped working in the game. But when I put the message boxes back in the program also didn’t work in the game, so there was a problem. It turned out that the problem was in the code of the game, The problem was that the game only registered when the button was going up, but we had functions activating when the button was down (pressed). After spending an hour on this problem we discussed about what button should be pressed for what action, and what commands we should have in our game (commands like attack, deflect etc.).

After this was done, I decided that the main program was done, I was still trying to create the function of users being able to make their own commands in a copy of the program, but that is something which I was unable to do. So I decided that that would come in a later update since I don’t have unlimited time for this project. And since everything was working I decided it was best to focus on what needed to be done instead of what would be nice to have.

Sources

On this page we have a list of sources we used for the research and the creation of the voice app.

* <https://docs.microsoft.com/en-us/archive/msdn-magazine/2014/december/voice-recognition-speech-recognition-with-net-desktop-applications>
* <https://developer.amazon.com/en-US/alexa/alexa-voice-service>
* [https://cloud.google.com/speech-to-text/](https://cloud.google.com/speech-to-text/?authuser=0)
* <https://azure.microsoft.com/en-us/services/cognitive-services/speech-to-text/>
* <https://dialogflow.com/>
* <https://www.ibm.com/cloud/watson-speech-to-text>
* <https://www.speechmatics.com/>
* <https://docs.microsoft.com/en-us/dotnet/api/system.windows.forms.sendkeys.send?view=netframework-4.8>