Gesture Based UI Development Project

|  |
| --- |
| Uamhan Mac Fhearghusa. (G00341809) |

## Introduction

The following document covers the speech recognition Tetris game found in this repository. From what purpose the application serves to the hardware used in the creation/runing of the software. The research behind the gestures chosen for this project and a full rundown of the architecture for the solution.

## Purpose of the application

For this application I decided to keep the user interface as simple as possible as the entire application is controlled entirely through speech. The main menu screen simply prompts the user to say start to begin the game. The purpose of this application to me was to learn more about speech recognition and its applications and how to tie it in with more traditional concepts. In this case how an old game such as Tetris would when you completely remove all user input other than speech.

## Gestures identified as appropriate for this application

The goal of this project was to learn as much about speech recognition as possible and as such the gesture controls are entirely speech recognition. The very nature of the game Tetris, quick reactions to the next shape and the spaces it can fit into. I felt this would make for an interesting challenge for the player, as it's one thing to quickly press buttons to control the piece how you want it. It requires a deeper level of concentration to be able to verbalize these commands. While the speed of speech varies a lot depending on the situation. According to the American National Center for Voice and Speech, the average conversation rate for English speakers in the United States is about 150 wpm. To contrast this the average typing speed is about 40 words per minute with regular users averaging 60+ words per minute this roughly translates to 200-300 letters or key presses a minute. Now when we replace a command in Tetris that was initially a key press, with a spoken word we can very quickly see that the user would be slower from the start (comparing 300 key presses vs 150 words per minute). This is not the only contributing factor in requiring greater focus using speech to control a Tetris game though. Most people that have played games for any extended period will develop a form of muscle memory to the actions. They will no longer think I need to move left, look at the keyboard press left, look back at the screen and then decide what to do next. They no longer need to think about the controls. They simply think I wish to move left and muscle memory dose the rest. Muscle memory is a very adaptable skill as discussed in the oxford university medium article on the amazing phenomenon of muscle memory. It being an adaptive skill ties heavily into playing games. As even if you have never played the game before chances are you have played something with a similar control scheme. As such you will have a good grasp for the controls right from the get-go. This ties in with our speech recognition controls. If you have never played Tetris before, but have played other games with 2d movement with the arrow keys muscle memory will kick in and you can focus on the game objectives rather than the controls. This is contrasted by the difficulty of verbalizing your given commands while still focusing on the gameplay not only because we naturally speak slower than press keys but also the muscle memory built up in playing with a keyboard. I believe this adds an interesting layer of difficulty/challenge or at the very least an interesting twist to an old game.

## Hardware used in creating the application.

As this project is based on Speech recognition for the gesture control aspect. The only physical hardware needed are a working microphone and a machine to run the software on. For the purposes of testing, the microphone I used was an audio Technica p48 studio microphone and a generic microphone found in a hp laptop. For the purposes of speech recognition controlling gameplay the difference of quality of mic made no apparent difference in the response time or accuracy of the speech recognition.

## Architecture for the solution

the full architecture for the solution, including the class diagrams, any data models, communications and distributed elements that you are creating. The architecture must make sense when the gestures and the hardware are combined. Justification is necessary in the documentation for this. You need to include a list of relevant libraries that you used in the project.

#### Overview/Libraries used

This Project was written entirely in the python programing language. I chose the python programing language as it is one of the languages that I am more confident in and I wanted to focus more on the speech recognition/game side of the project than getting caught up in program language specific problems. As I was recreating the old game of Tetris, I chose to implement it using the pygame library. It provides robust and easy to use GUI architecture to base the game off of. The Speech recognition library was used as an api to communicate with the google text to speech engine. The random library was also used to randomly select from the possible pieces. The python threading library was also used so we could have the speech recognition section of the program running on one thread while the main game loop runs on another allowing the game loop to continue running while waiting for the speech recognition. The Re library was used for regular expressions. These regular expressions where used to give leeway between what the user says and the controls of the game for example they were used in the control for moving a piece left if the recognized word from the users speech started with an le or ended with an ft it would count as the user had said left. This allowed for some of the inaccuracies for the speech recognition and the mispronunciation of words from the user in the heat of playing the game.

Librarys Used.

Pygame. <https://www.pygame.org>

Speech\_recognition. <https://pypi.org/project/SpeechRecognition>

Threading. <https://docs.python.org/3/library/threading.html>

Re . <https://docs.python.org/3/library/re.html>

#### Main architecture

This program consists of two main components. The Game loop and the speech recognition. The game loop consists of each game tick the current block moving down one space checking user input to move the block. Checking if the block hits the bottom or if the player has lost. This is achieved with various method calls from the main game loop. Along with gui method calls to draw the relevant information to the screen. We keep static variables of Gui parameters such as screen width and the grid size for the game screen etc. These are used within the GUI methods to ensure the same size is drawn to the screen each game tick. The potential pieces that can appear are randomly selected from a list of possible pieces whenever a piece hits the bottom of the game grid. Only one class needed to be created for this architecture and that was the Piece class representing the game piece.

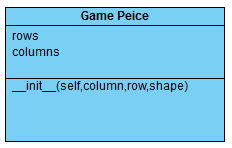
The speech recognition part of the architecture contains a next\_comand variable which is also used in the main game loop as the next user input to control the current piece. The speech recognition section initializes the speech recognition connection to google speech recognition engine and initializes the systems default microphone as the microphone as the input to listen to for commands. within the main loop for audio recognition It adjusts for ambient noise for more accurate results from the microphone then records the user input and stores it in an audio variable this audio variable is then converted to a string of the word spoken. The next command variable is then set to this string.

If these two loops were run concurrently the game loop would stall while waiting for the user input which would interrupt the game loop to prevent this, I decided to apply threading. The main game loop and the speech recognition loop are both run on different threads to allow the game loop to keep running while waiting for speech input.

#### Controls

The controls are implemented through the next command variable. Each game loop iteration, the loop checks if the next\_comand variable has a valid command as a string using regular expressions for leeway in the user input. If the user says left the game piece will move left and right if the user says right, it will drop faster if the user says down. To rotate the game piece the user says turn. I chose the word turn instead rotate as I felt turn was a quicker and more smooth command than rotate.

#### Class Diagrams



## Conclusions & Recommendations

From undertaking this project, I learned a great deal on the topic of speech recognition and its strengths and limitations. The accuracy of the speech recognition was far better than I initially thought it would be. The short comings were in the time it takes to get the results. It took on average 2 seconds to get a text response of a command and while in some implementations this could be perfectly acceptable in a fast-paced real time game such as Tetris this delay made it difficult to play the game at a high standard this was offset by slowing the game speed down and allowing for less accurate results through regular expressions. If I were to attempt this project again, I would look further into how to reduce this delay. One possible solution might be to allow for a constant audio stream to the speech recognition engine and return a command every time a pause in input is detected.

## References

***National center for voice and speech.***

<http://www.ncvs.org/>

***Oxford university article on the amazing phenomenon of muscle memory.***

<https://medium.com/oxford-university/the-amazing-phenomenon-of-muscle-memory-fb1cc4c4726>

***Wiki link for speech recognition for general overview***

<https://en.wikipedia.org/wiki/Speech_recognition>

#### Pygame library

<https://www.pygame.org>

***Speech\_recognition library.***

[ttps://pypi.org/project/SpeechRecognition](https://pypi.org/project/SpeechRecognition)

***Threading documentation.***

[ttps://docs.python.org/3/library/threading.html](https://docs.python.org/3/library/threading.html)

***Re regular expression library.***

<https://docs.python.org/3/library/re.html>