Journal of Fundamental and Applied Sciences

ISSN 1112-9867

Available online at http://www.jfas.info

Research Article
Special Issue

REAL TIME PITCH DETECTION GAME TO INTRODUCE LOCAL SONGS

M. Puspitasari[*], M. Safrodin, F. N. C. Bagar

Department of Creative Multimedia Technology Electronic Engineering Polytechnic Institute of Surabaya, EEPIS Surabaya, Indonesia

Published online: 01 February 2018

ABSTRACT

Nowadays, Indonesians are getting less interested in traditional songs. This leads to more of the younger generation being unfamiliar with their own local songs. This research aims to reintroduce local songs, especially to younger generation with game as a media. Inspired by karaoke box in which user sings along with the music, players in this game are able to control their character using their voice. The character they're playing will move up and down based on how high or low the pitch they sing. In the other hand, the obstacles in this game is also automatically generated by the game following the melody of the song played in the background. Applied in this research are the FFT theorem and Blackman Harris windowing technique. The combination of both theorem and technique are implemented in Unity game engine with C# as programming language. The unique game play is implemented in order to gain interest of public, so that the goal of the research could be met.

Keywords: local song; pitch detection; FFT; Blackman Harris

1. INTRODUCTION

Indonesians' interest in traditional songs is in decline. Younger generation is especially unfamiliar with the songs as they're not as popular as other songs. This is very unfortunate, because, the content of local songs is really good for mentality education.

Author Correspondence, e-mail: mastutipuspitasari@gt.student.pens.ac.id

doi: http://dx.doi.org/10.4314/jfas.v10i2s.24

In the other hand, following the vast development in technology, people, especially kids and teenagers, are mostly interested in modern entertainment such as video game. This is the reason why we chose to use video game as a media in order to solve the problem in question.

The game developed in this research, "URA-URA NUSANTARA", is named after the word *ura-ura*, which means "songs" in Sansekerta language, one of the oldest languages that used to be spoken in Indonesia, and the word *nusantara*, which is one of alternate names of Indonesia back in the past. This game is mainly inspired by *karaoke* games and Flappy Bird in terms of gameplay. *Karaoke* games, such as Singstar and Guitar Hero Vocal, challenge players to sing along and match their pitch to their markers in order to get the highest point, while in Flappy Bird, players are challenged to go through unlimited series of obstacles on top and bottom part of the screen. In this game, player sings along with the music, just like in *karaoke* box, but their voice will control a character based on their pitch, moving it up and down through a series of obstacles, similar to Flappy Bird. Score will then be calculated after a session of one song, based on the player's performance in avoiding the obstacles by singing in correct tune. So, in this game, the obstacles are limited and hitting the obstacles won't kill the character. Instead, it would result in less score at the end of the game. Unlike rhythm games in general, the obstacles generated in this game are automatically spawned based on the songs MIDI in real time. This is done to make sure that the obstacles are placed accurately according to the song's melody and ease the process of level design, in present and future, as we will only need to replicate the MIDI instead of manually placing the obstacles for every time a new song is added to the game.

Character control and obstacles generation are both done in real time, using microphone to take player voice as in input and MIDI of the song for the latter. In game development world, pitch based character control and obstacle generation are still very uncommon. Hopefully, its uniqueness could encourage people to play the game, which would mean more exposure and less likeliness of the extinction of Indonesian traditional songs included in the game.

2. BASIC THEORY

To develop this game, general knowledge related to pitch detection is needed. It will then be implemented in the game only as a part of the system. Below are the topics needed to be covered in order to grasp the idea of how pitch detection is done:

2.1. Definition of pitch

In musical context, pitch is one of the features of an audio signal that is relevant, only when human is present as the receiver of the signal. Despite the fact that it is measured and classified based of the frequency of a sound, the concept of tune classification based on its pitch was already introduced long before spectral content was understood in science. Musicians at that time only depended on their hearing ability to differentiate and classify tunes with no specific formula. The value of pitch is related to log value of a tune frequency. This is proven by the fact that one octave difference is present when a tune is scaled by two times in its frequency, with small deviation in tune with frequency less than 1000 or more than 5000 hertz. In lower frequency, which is less than 1000 hertz, frequency multiplication by two results in change that is less than an octave, while the opposite is true in frequency more than 5000 hertz. On a side note, this calculation was done in isolated sinusoids, which are less complicated compared to real life sound spectrum, while pitch perception itself is affected by complexity of the signal that includes the intensity, duration, and other physical features of the sound wave [1].

2.2. Pitch detection

Also known as fundamental frequency estimation, this topic has been quite popular in research until now. The problem explored in this topic is basically the process of extracting fundamental frequency value from a sound signal component that has the lowest frequency or harmonically related to the other components. The frequency of this component is what we call as fundamental frequency. To do the extraction, there are several standard methods used by researchers. Using these methods, it is possible to extract the fundamental frequency from a sinusoid consisting of a sound (Fig. 1a). However, when a sinusoid consists of two or more sounds, the process becomes more complicated, as we need to break the sinusoid to its components and search for the fundamental frequency which could still be easily detected (Fig. 1b), barely detected (Fig. 1c), or not detected at all (Fig. 1d), depending on how strong the harmonics are[1].

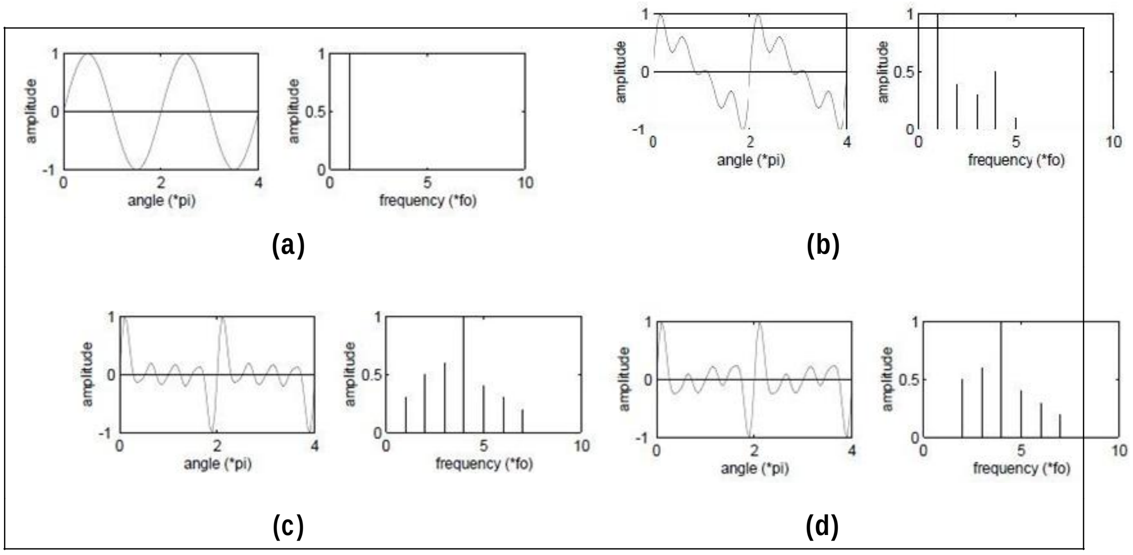

Fig.1. Effects of upper harmonics in pitch extraction [1]

2.3. Fast fouriertrans form

Signal in real life is usually represented by voltage that changes over time, which is also known as time domain. Fourier theorem explains that every waveform present in time domain can be represented by the weighted sum of sines and cosines [2]. Below is an example of two simple waveforms in one time domain and a combination of both:

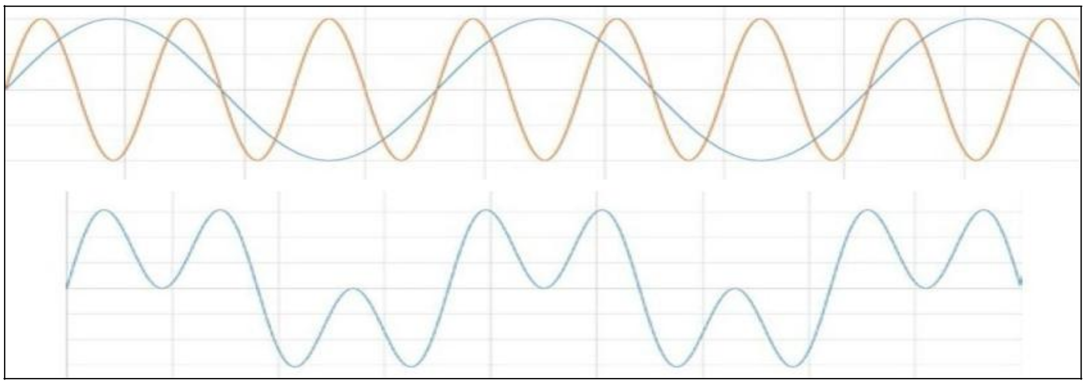

Fig.2. A new signal is formed by adding two signals [2]

This is also true in reverse; a complex signal may be broken down to its components, which allows us to analyze each frequency in the signal. Fourier transform deconstructs a time domain to a frequency domain representation as shown in Fig. 1 (time domain on the left and frequency domain on the right side). The amplitude, which is shown in both domain is important in defining the original sine wave, which has the biggest amplitude value, while the other frequency with lower amplitude can be ignored in process [2].

2.3. Windowing

FFT calculation to break down a signal to its components frequency is based on circular topology, which means that the start and the end of the signal has the same value. However, a signal may not have integer number of period which leads to miscalculation in FFT. Known as spectral leakage, this causes the fine spectral lines to spread into wider signals[2].

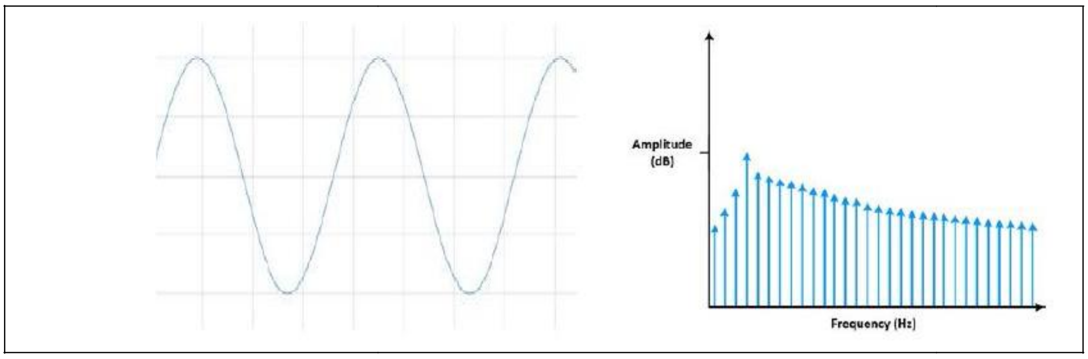

Fig.3. FFT implementation in a signal with non-integer period [2]

To minimize the effect, a windowing technique can be applied. This technique decrease the amplitudes on both ends of a signal in time domain representation by multiplying a continuous sine function that scales from zero to highest amplitude back to zero. This way, the significant difference of both ends could be omitted[2].

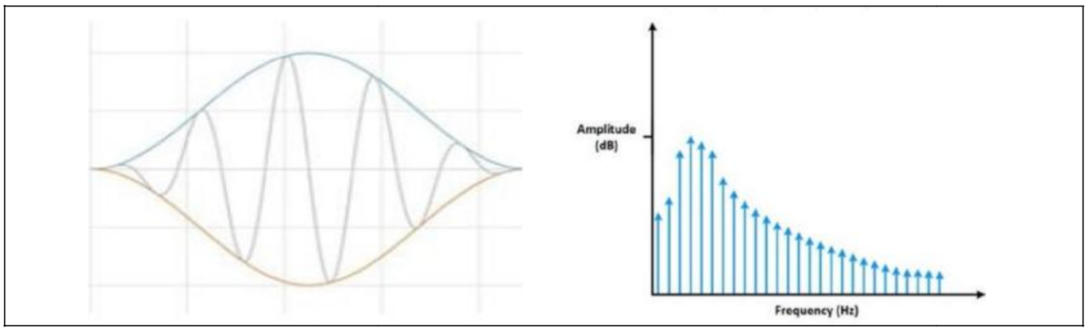

Fig.4. Windowing technique implementation [2]

There are many kinds of windowing function that can be applied. It is necessary to understand the characteristics of windows in order to decide which one is the best to implement for each case.

Blackman Harris, for example, has a good side lobe compression [2] and is used by many for pitch detection.

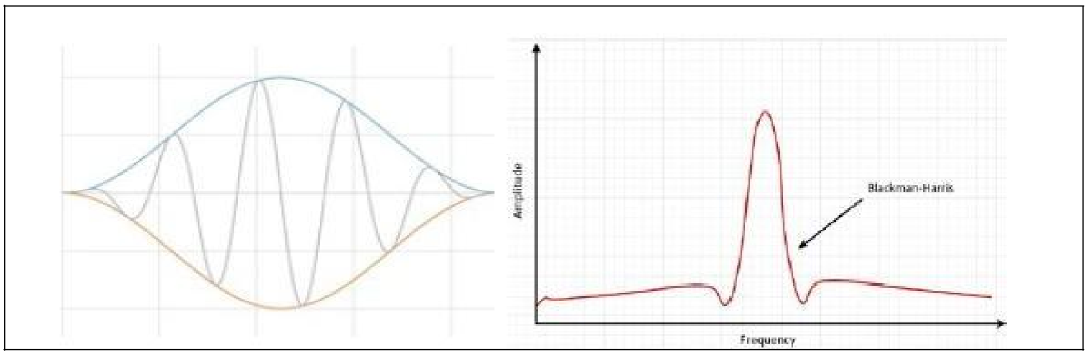

Fig.5. Blackman Harris window implementation [2]

3. CREATING THE GAME

The game needs to be carefully designed in order to ensure that players would give it a try. This includes the design of the game flow, visualization, scoring, features, and of course, the implementation of real time pitch detection itself and its role in the game. The development of the game is done in Unity game engine, using C# as the programming language, while its visual and audio assets are created using Inkscape and FL Studio respectively.

3.1. Game flow

The game starts with a splash screen that’s followed by main menu. Here, a player can choose to play, practice, or delete the score record. When a player chooses to play, they will be brought to song selection screen that will show a map of Indonesia with several dots on the map, each representing a traditional songs. After a song and difficulty are chosen, a session of one chosen song will be played, followed by their score at the end of the session. They will then be able to choose whether to replay the song or go back to the song selection screen. During a session players can also pause the game and choose to continue, restart, and quit the song. In practice menu, players can spawn random obstacles and try to guess what tone should be sung in order to avoid hitting them. The score is also shown in this menu. This way, they can learn what tone to sing according to the obstacle’s height. Lastly, if they ever want to delete the score record, they can access the delete menu and the system will erase all high scores of the songs listed in the game.

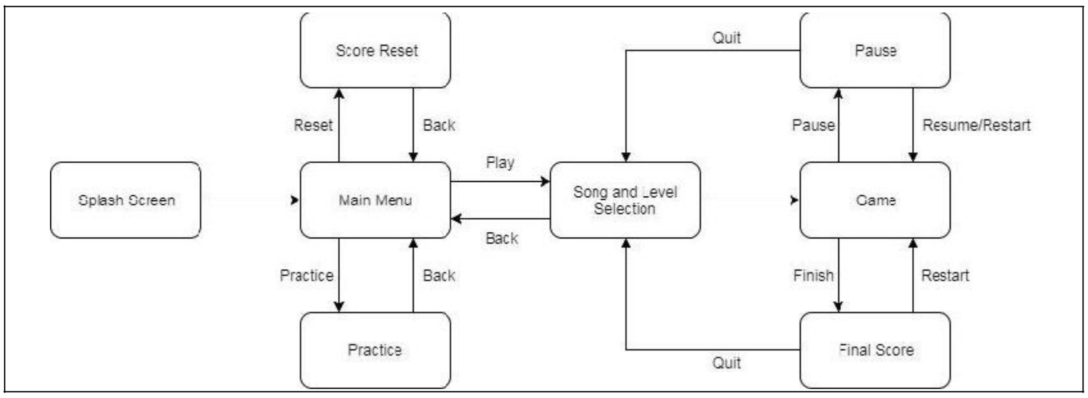

Fig.6. Game flow

3.2. Core mechanics

Character control in this game utilizes microphone to receive player voice as input, while the obstacles are automatically generated from vocal part of the song, also known as melody, that is represented in MIDI form. MIDI is chosen because it is more stable and easier to extract pitch from, since the system isn't optimized in any way. Both voice and MIDI are processed in real time and their pitch are extracted, each with their own role in the game. Pitch of player voice will be used to determine flying height of the character, Ura, in the screen, while pitch of song MIDI will be used to determine top and bottom obstacles' position through a game session. The height of both character and obstacles are calculated by remapping detected pitch to one of the registered positions that are vertically mapped from top to bottom part of the screen. The higher a detected pitch is, the higher Ura flies or an obstacle is placed. While Ura movement is limited on screen, top or bottom obstacle may not be visible when an extremely high or low pitch is detected.

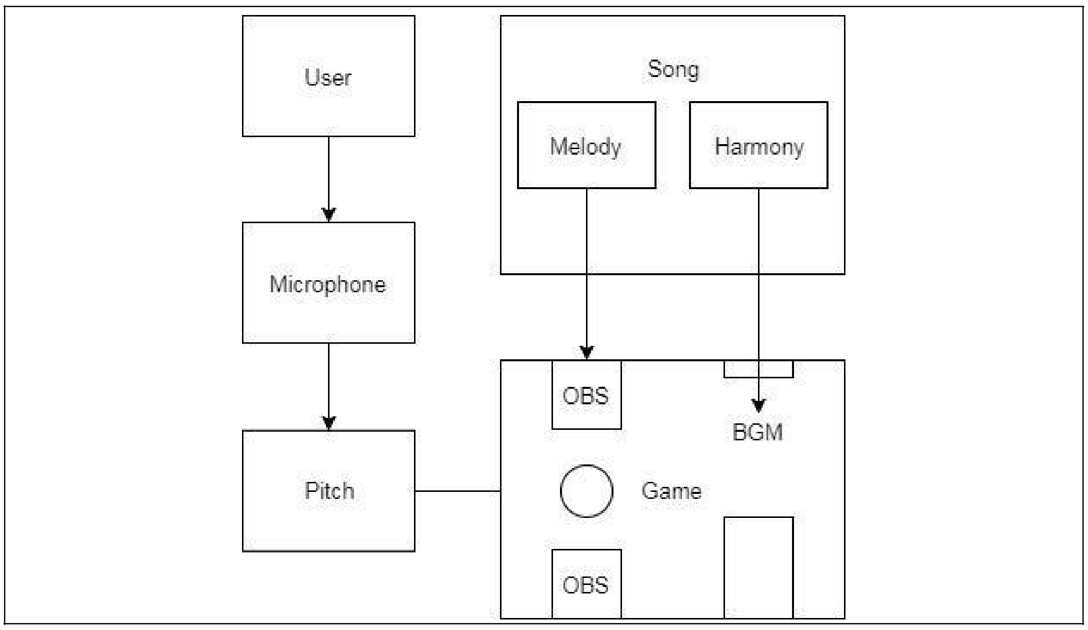

Fig.7. Core mechanics

To register microphone as an input in Unity, we have modified the script from reference [3] to meet our necessity. Song in the games will be reconstructed into two parts, melody and harmony. This reconstruction is based on reference [4] which claims that separating melody from the harmony of a song would ease the pitch detection process. Microphone input and melody will then be processed through pitch detection algorithm which we took and modified from reference [5] and converted into height of the character and obstacles respectively.

3.3. Scoring

Player performance is scored according to the how good they are in avoiding the obstacles. As the system only detects pitch, it is not necessary for player to sing the lyrics correctly, they may even use musical instrument to play the game, but knowledge of what tune they are supposed to play is still needed in order to avoid hitting the obstacles and get the highest possible score, which is 100 points. Each time Ura goes past an obstacle, its position will be evaluated and classified into either of these four categories: perfect, great, good, or miss, depending to Ura's distance to both top and bottom obstacles. The closer it is to any of the obstacles will give less score and vice versa, so what the player needs to do is basically balancing Ura position in the middle of both obstacles. Below is the calculation of final score of a gamesession:

$$score = (A + 0.7B + 0.3C) / T * 100 \qquad \text{……….(1)}$$

Where A, B, C, and T respectively represent perfect, great, good, and total obstacle counts. This means that every perfect, great, good, and miss, grant the player 1, 0.7, 0.3, and 0 point respectively, while the value of a point depends on how many obstacles are present in the song played. Here's an illustration of each point's area in the game:

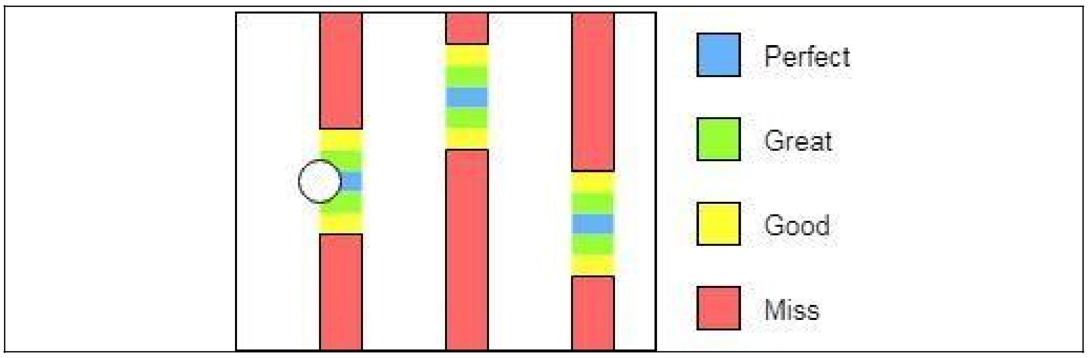

Fig.8. Scoring

4. EXPERIMENTS AND ANALYSIS

To test the performance of pitch detection in the game, we conducted two kinds of experiment, the first using MIDI sample and the latter is done live using a piano application.

4.1. MIDI sample test

For this experiment, MIDI sample is tested. The sample was made based on Indonesian traditional song, "Bungong Jeumpa". The range from G3 to E5 [6], including only the flat notes, is used as current limitation in the game. The experiment is done by recording detected pitch through a session and comparing the result with melody of the song represented in numbers from -2 (G3 equivalent) to 10 (E5 equivalent). The recording of detected pitch itself is done by utilizing the script found in reference [7] with modification, which allows us to record the data in txt file. The data are then copied and pasted in a spreadsheet to be compared to the reconstruction of the song sheet. Below is the result of the experiment:

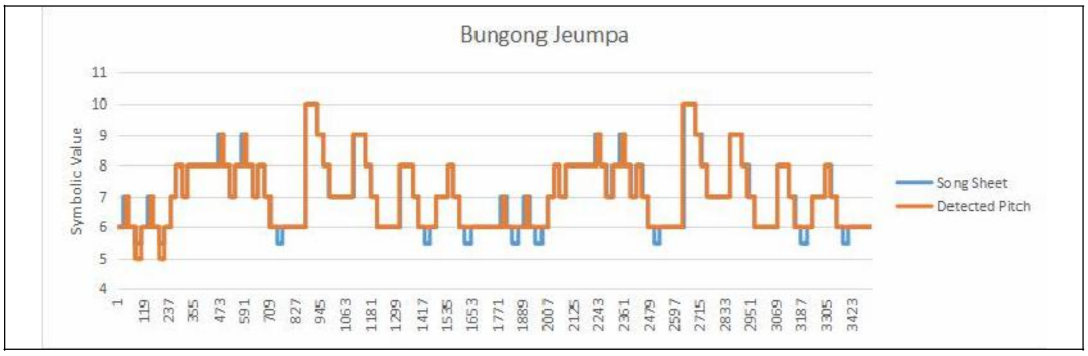

Fig.9. MIDI experiment result

As seen from above, the system fails to detect G4# (5.5 equivalent) and would sometimes detect a note either earlier or later than it should be. G4# is still excluded from the pitch list, as the system is still limited to detect only flat notes, while the difference in detection time itself is cause by the difference in tempo during MIDI composing. In order to improve the system, semitones may be added in future development of the game, so that the notes such as G4# can be detected in the game.

4.2. Live piano test

This experiment is conducted to make sure that the system can distinguish fundamental frequency from background noise. In this experiment, white piano keys will be pressed from G3 to E5 each for a brief moment so the graph is expected to show a gradual increase in value. A piano is chosen over real human voice due to its stable pitch which ensures no debate whether the tone is correct or not. The application used in this experiment is Piano HD available on iPad, using marimba instruments, while the microphone used is a built-in on ASUS X550D laptop. The system has an amplitude threshold included to omit background noise. However, this also means that input has to be loud enough to pass the threshold.

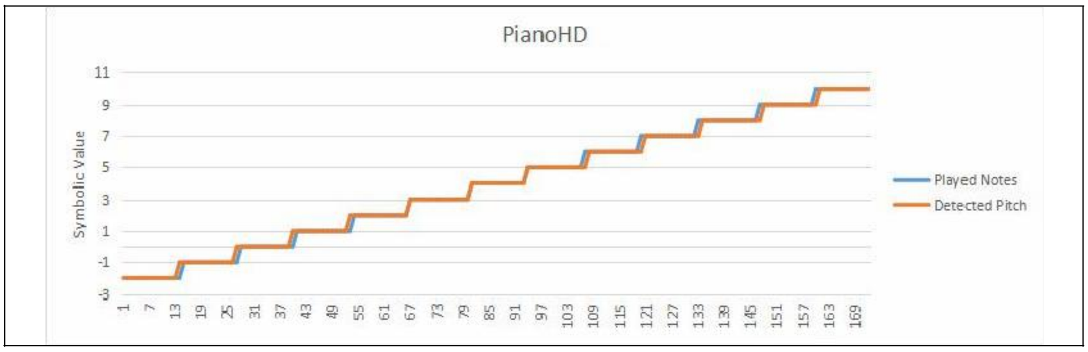

Fig.10. live piano experiment result

This experiment was done in a quiet room to minimize chance of loud background noise and the graph shows that every note played live on Piano HD was detected correctly. However, please note that presence of background noise that is stronger than the input may affect the reading, resulting in wrong pitch detection result.

5. CONCLUSION

Traditional songs in Indonesia are getting forgotten day by day. This situation calls for a better approach in keeping the songs alive in recent days. Following modern trend, video game is thought to be a potential media to reintroduce local songs. After studying several music games as reference, this game was developed, first with three local songs and basic features included. According to conducted experiments, the core function of this game is already working as designed. However, the system itself is still very potential to be enhanced, especially in terms of features. When the game is published, we hope that it could serve as a solution for the problem, so that local songs are more exposed to younger generation.

REFERENCES

[1] David Gerhard, “Pitch Extraction and Fundamental Frequency: History and Current Technique”, Department of Computer Science, University of Regina, Canada**(2003)**

[2] “Instrument Fundamentals: Understanding FFTs and Windowing” (white paper), National Instruments**(2017)**

[3] Megan,“HowDoIGetUnityToPlaybackAMicrophoneInputInRealTime?”:https://support.unity3d.com/ hc/en-us/articles/206485253-How-do-I-get-Unity-to-playback-a-Microphone-input-in-real-time-**(2017)**

[4] Chao-Ling Hsu, DeLiang Wang, dan Jyh-Shing Roger Jang, “A Trend Estimation Algorithm for Singing Pitch Detection in Musical Recordings”, IEEE ICASSP**(2011)**

[5] Aldo Naletto, “GetOutputData and GetSpectrumData, what represent the values returned?”: http://answers.unity3d.com/questions/157940/getoutputdata-and-getspectrumdata-they-represent-t.html**(2011)**

[6] B. H. Suits, “Frequencies for equal-tempered scale, A4=440 Hertz”, PhysicsDepartment, Michigan Technological University: http://pages.mtu.edu/~suits/notefreqs.html**(1998)**

[7] “File.WriteAllText Method (String, String)”, Microsoft API and reference catalog.

Microsoft: https://msdn.microsoft.com/en-us/library/ms143375(v=vs.110).aspx**(2017)**

How to cite this article:
Puspitasari M, Safrodin M, CahyaBagar F N. Real time pitch detection game to introduce local songs. J. Fundam. Appl. Sci., 2018, *10(2S)*, *311-322*.