DiaTones: Sight-Sing Sheet Music to Win

Peter Tessier
fpv5gr@virginia.edu
University of Virginia
Charlottesville, Virginia, USA

Figure 1: Sample Level of the DiaTones Game

Abstract

The ability to sight-sing music is important for singers, especially in choral ensembles. Unfortunately, many obstacles hinder singers' ability to learn this skill: lack of motivation, lack of classroom time, inadequate feedback, and instructors' own difficulty with sight-singing. Past work has shown that gamifying the sight-singing process mitigates these problems: singers are more motivated, are more likely to practice outside of class, receive immediate visual feedback, and do not need skilled supervision. Hence, we developed the video game *DiaTones* featuring real-time pitch detection, dynamic musical score display, and engaging visuals to improve sight-singing skills in a fun way. Such a game differentiates itself from previous aural training software and video games by possessing all three of the following: strong gamification elements, real-time visual feedback, and sheet music - not a midi track - as a medium for showing notes. In evaluation by playtesting, the game showed potential to both improve sight-singing ability and keep players engaged for long periods of time. In the end, *DiaTones* finds a useful niche among vocal training software, providing one more tool to help teach the important skill of sight-singing.

Keywords

Video game, Sheet music, Sight-reading, Sight-singing, Pitch detection

1 Introduction

The importance of sight-singing (sometimes also referred to as sight-reading) is emphasized throughout literature [1, 14]. In his sight-singing compendium, Demorest opens with "There is almost universal agreement on the importance of teaching musical literacy as a means to musical independence" (page 1) [9]. He also mentions how the 1994 Congressional National Standards for Arts Education states sight-singing as an achievement standard for grades 5-12.

Despite the widely-recognized importance of sight-singing, Demorest laments how several obstacles prevent instructors from putting this learning to practice in the classroom: they feel not skilled enough, they fear it is boring for the students, and limited class time is spent almost exclusively for rehearsing performances. In particular, other authors concur that lack of motivation [20] and lack of classroom time [14] hinder students' ability to learn this vital skill. Paney adds that the lack of real-time feedback makes it harder for students to learn from their mistakes, as pitch inaccuracies are typically pointed out well after the fact (e.g., by the instructor after the piece has been recited).

To address these issues, we developed *DiaTones*, a piece of software that gamifies the sight-singing process to supercharge motivation, empower singers to train in their own time, and provide valuable visual pitch feedback. Indeed, past studies have shown that video games and other software tools are better at improving sight-singing skills than traditional methods [14, 20]. Already, there is an abundance of existing software which already supports this purpose, such as karaoke games, sheet music games, and automated tools for grading aural exercises. However, *DiaTones* stands out because unlike the breadth of existing sight-singing software found [2–6, 10, 12, 15–17], it is the first to combine the aspects of strong gamification, continuous pitch visualization, and sheet music. This gives it a unique edge in motivating students to practice sight-singing of real sheet music.

2 Related Work

There are three main categories of existing work which is similar to *DiaTones*: (a) karaoke games, (b) sheet music scoring software, and (c) pedagogical tools solely meant for instructors with students.

In the first category, a plethora of examples can be found: *Rock Band* (a series with four main games and over $1B in sales) [6], *Let's Sing* (a series with at least 18 titles from 2012 to 2025) [3], *SingStar* (a 1-v-1 PlayStation series) [12], *Karaoke Revolution* (an earlier game from the studio Harmonix, behind Rock Band) [2], and *Midi Karaoke* (a free online karaoke game that generates a level based on any midi file) [5]. Karaoke games tend to create fun by embracing the "fantasy" aesthetic as described in the Mechanics-Dynamics-Aesthetics (MDA) Framework [13]; i.e., the player enjoys it because they feel like they are a famous/talented singer, with strong, flashy visuals to support that fantasy. While these games do feature prominent gamification elements and real-time pitch detection (in every case, via a sprite like an arrow that moves vertically based on detected pitch), they always use scrolling midi notes to show the player where the pitch should be. While this still helps users improve intonation [19], it fails to train them on actual sheet music.

In the second category of sheet music scoring software, at least two examples can be found: the mobile app *Sight Singing Pro - Solfege* [15] and the all-around aural training desktop application *EarMaster* [10]. These are probably the closest to *DiaTones*, since they feature pitch detection and use sheet music (not midi notes); however, their interfaces resemble an educational tool more than a video game, and their pitch-detection is not really in real-time, instead showing results after-the-fact. This limits the amount of fun (and hence motivation) they provide, as well as the usefulness of intonation feedback.

Finally, the third category of school-exclusive tools include *EarTrainer.io* (which automates grading for aural assessments but excludes pitch detection) [17] and *Make Music Cloud* (which similarly allows teachers to assign a range of aural assessments, also featuring pitch detection for grading) [4]. Like the previous category, these tools lack the "fun" of true video games and have the additional disadvantage of only being accessible to singers enrolled in institutions that have subscribed to these services.

There are a couple other pieces of software that inspired *DiaTones* in very different ways. First, *Sight Reading Factory* provides singers with a limitless amount of parameterizable sight-reading samples that are especially helpful in preparing for standard sight-singing auditions [16]. It is considered the standard for sight-reading music, even if it does not support scoring the user based on pitch detection. The other software is *EpiStory - Typing Chronicles*, which teaches the skill of keyboard typing via a story-rich, exploration-based video game that features satisfying elements of progression like leveling up, stat increases, and skill acquisition [8]. If *DiaTones* were to be expanded, it would take inspiration from this model to embed itself into an adventure game with a world and story to follow. Hence, by appealing to the additional aesthetics of discovery, narrative, and fantasy, players' motivation would increase further.

3 Technical Considerations

The *DiaTones* game itself involves attempting to reach as high a score as possible in various "levels." Each level has a musical staff with information such as the clef, key signature, and time signature. Musical notes scroll from right to left, and the user is to sing the correct pitch once each note crosses a vertical line on the left. Such a system required solving a number of non-trivial problems, with solutions described here.

3.1 Implementation Details

The game was implemented in the open-source video game engine Godot. To explain how it works, the developer can define any number of "scenes" which have a hierarchy of "nodes", which can be thought of as reusable classes with functionality specific to their type. Godot was chosen due to its ease-of-use, generous licensing, and support of custom nodes built from C++ (allowing for fast pitch detection).

3.2 Real-Time Monophonic Pitch Detection

A fundamental requirement of *DiaTones* is the ability to take continuous streams of audio data from the microphone and extract the pitch. In general, this problem of pitch detection is a challenging one that often requires re-engineering for every different application. It is so difficult because there is so much data to be processed so quickly: in this case, 2048 samples of microphone input every 10 seconds. Moreover, the naive approach for detecting the pitch is a well-known algorithm called autocorrelation, which - to oversimplify - exploits the fact that if one shifts a waveform by x and gets roughly the same waveform back, then that waveform must have a pitch of x. The problem is, such an algorithm runs in $O(n^2)$ time for n samples, which means there are an order of $2048^2 \cdot 10 = 41,943,040$ multiplications that must happen every second - far too slow.

A more clever way to find pitch invokes the Fast Fourier Transform (FFT), which brings down the time complexity to $O(n \log n)$, significantly reducing the number of multiplications per second to the order of $2048 \cdot \log_2(2048) \cdot 10 = 225,280$, an incredible 99.5% decrease. In fact, such a method which uses the FFT and provides a number of other useful optimizations for real-time monophonic pitch detection (such as peak-picking) is the McLeod Method [18], which was used in the past for the super-accurate Midi Karaoke (we contacted to developer to confirm), and is what we ended up implementing in C++ for this project. For comparison, when testing the frames per second (FPS) using both models, naive autocorrelation was unable to perform higher than 40 FPS, while the McLeod Method never dipped below 60 FPS. In the end, we chose to only run the pitch detection 10 times per second anyway since such computation is still quite expensive, and no different is observable from the user perspective.

3.3 Displaying Pitch on a Musical Staff

As we attempted to code the actual display of the pitch being detected on the musical staff, one technical consideration which karaoke games did not have to worry about quickly became apparent: unlike on a midi track, if one walks x amount up from a given position on the musical staff, the pitch traveled is not constant. For instance, in the key of C Major, the distance between E

and F is 1 semitone, and the distance between F and G is 2 semitones; however, both of these distances on the physical staff are the same. This means that if the detected pitch is moving at a constant speed from E to F to G, it must be shown as traveling twice as fast when between E and F compared to F and G. Moreover, if the key is instead G Major, then a detected pitch of F natural should be displayed slightly lower than it would be in the key of C, since now that note is too flat (the note that should be sung is F sharp, since G Major has that note). In general, the key affects how the pitches are displayed.

To account for the non-linear scale and dynamic keys, a mathematical formula that perfectly converted midi notes into location on the staff was derived and tested visually on the graphing calculator Desmos.

3.4 Generating Levels from MusicXML

An interesting problem was how to generate levels. Fundamentally, this involves telling the game the clef (which is different based on voice part), key signature, time signature, beats per minute (BPM), and every individual note/rest. One solution would be to create some representation of such information by hand, with each note having information about whether it is a rest, its pitch, accidental information (e.g. sharp or flat), its duration (e.g. half or quarter note), and where in the song it played (e.g. at measure 2, beat 3). Such a process was attempted, but proved to be tedious. Instead, a more streamlined approach was adopted by using the free music notation software MuseScore to create sheet music, then export the music as an xml file which was then parsed by a node in Godot, producing the representation object automatically. From the xml file, the key signature, time signature, BPM, and note data is obtained, updating how the staff appears. With such an approach, making a new level is as easy as telling Godot which xml file to parse and which mp3 file to play in the background while the user sings. That being said, functionality such as complex rhythms (like triplets), multiple voices, and key/time signature changes are not yet supported. In the future, a goal would be to expand the xml parsing expand support for such features, both to enable to developer to create more complex levels, and also give users the ability to generate a custom level from a wider variety of MusicXML files.

3.5 Synchronization with the Beat

Syncing the gameplay with audio is a recognized challenge for the Godot engine [7]. Fundamentally, there must be some latency between asking Godot to play a sound and that sound actually playing on the speakers. We followed official documentation to account for this latency via provided functions inside a "Conductor" node. This node is then responsible for keeping track of the precise time since the start of the song and emitting a "beat_hit" signal whenever a beat hits, based on the BPM (e.g., for 60 BPM, the Conductor would emit "beat_hit" once a second). Finally, this signal is observed by other nodes, such as the NoteGenerator which uses beat information to animate the notes across the screen.

3.6 Scoring

The final challenge involved deciding how to assign a score to the player's performance and how to communicate this visually. A relatively straightforward approach was ultimately taken: for every note in the song, the game detects how far away the detected pitch is from the target pitch. To be somewhat forgiving, any pitch within 30 cents of the perfect pitch was given a score of 100%. Any pitch 60 cents or more away from the target pitch was given a score of 0%, and a linear scale was applied for any pitch in between (so, for instance, a pitch 45 cents away was given a score of 50%). At the end of the song, all of these note-scores are aggregated into a final score for the song, and the final score out of 100 is displayed to the player. To give the player feedback while singing, a "pitch meter" shows the player how flat/sharp their singing is relative to the target pitch, helping them correct their intonation in real time.

One notable obstacle in making accurate, responsive scoring was accounting for the latency in pitch detection. Indeed, even with the optimized pitch detection algorithm, some delay between the user singing the note and the game determining the pitch is unavoidable, meaning the scoring would always be a bit behind, incorrectly marking notes as sung wrong. In the future, a calibration feature that lets the scoring account for this latency would be one of the most importance fixes to address this issue.

4 Evaluation

To evaluate the success of *DiaTones* based on its goal of being a fun way to practice sight-singing, we were interested in two broad questions:

(1) Does *DiaTones* help users to sight-sing better?
(2) Is *DiaTones* fun/enjoyable/usable?

An informal case study was performed to explore these two questions by playtesting the game on a handful of a cappella members. We made the following observations.

4.1 *DiaTones* has Potential for Sight-Singing Improvement

By seeing real-time feedback, users were able to correct their pitch based on the feedback they were viewing. It was observed that users would tend to sing more off-pitch at the beginning, then tend to have higher scores near the end. It is unclear how much of this was a result of learning the *DiaTones* system itself versus actually improving pitch detection, but this provides some evidence that *DiaTones* in fact improves sight-singing ability. Additionally, users reported the real-time pitch feedback as helpful, some surprised that their pitch wasn't as accurate as they thought it was. Further research should be done to compare *DiaTones* to more traditional methods - such as using Sight Reading Factory - to greater validate its efficacy as a sight-singing tool.

4.2 *DiaTones* is Generally Engaging

More significantly, *DiaTones* across the board kept players engaged for the several songs they playtested. The positive visual feedback proved to be a powerful motivator, encouraging players to continue singing the correct notes and improve as their score increased. Indeed, the gamification elements of an increasing score and flashy colors seemed to provide the much-needed motivation for sight-singing, lacking in other tools.

5 Remaining Challenges

Though *DiaTones* is successful at providing a fun way to practice sight-singing, there are several important improvements to address in the future.

5.1 Improving Pitch Detection

As stated previously, the McLeod Method was used to detect pitch in *DiaTones*. This worked relatively well, but still maintains some occasional inaccuracies - especially with fast notes and when changing pitch - as well as about a tenth of a second of latency. There are several other methods which have not been explored as a replacement to the McLeod Method in *DiaTones*, but future research could see if these serve as an improvement. In particular, Faghih found that YinFFT and Fast Yin were the two highest-scoring algorithms in his survey. His results also highlight the yields to accuracy that post-processing can do, such as dynamic pitch-shifting and changing audio buffer size based on vocal range [11]. These post-processing techniques can also be explored.

5.2 Improving Scoring

As mentioned earlier, the latency of pitch detection makes the scoring inaccurate. One solution to this which should be explored is calibrating the game to this latency, which is a common approach in rhythm-based games. Additionally, a "streak" of correct notes can be explored to be compounded into the score as a way of making the score more forgiving and dynamic while also adding another layer of fun and motivation.

5.3 Increasing Fun/Motivation

Due to the success of the level format, we would like to explore the possibility of embedded such levels as encounters in an adventure/RPG game which resembles the exemplary EpiStory. Such a fleshed-out immersive video game experience would supercharge fun and motivation by leaning into more MDA aesthetics, most prominently discovery, fantasy, and narrative.

5.4 Improved Evaluation

Like a longitudinal study recommended by [19]; e.g., provide the same participants with *DiaTones* over a period of a few weeks and compare their sight-singing level before and after versus a control group. This would especially work well since the idea of *DiaTones* is that with higher motivation, users will choose to practice it more on their own than traditional practice.

6 Conclusion

DiaTones was created to supplement the existing body of software tools aimed at helping singers improve sight-reading skills. This video game distinguishes itself among other tools by featuring all three: immediate pitch feedback, sheet music, and prominent gamification. In this way, it has the unique advantage of providing strong motivation to practice sight-singing, an essential area of musicianship. We found that *DiaTones* showed promise in being both effective at improving sight-singing skills and fun/engaging. With such a tool, singers can have a better time training this essential skill.

Acknowledgments

To Prof. Mark Sherriff for his advising

References

[1] 2014. *How Important is Sight-Singing?* https://chorusamerica.org/singers/how-important-sight-singing
[2] 2003. *Karaoke Revolution.* https://www.imdb.com/title/tt0386569/ IMDb ID: tt0386569 event-location: United States.
[3] 2025. *Let's Sing 2025 - The ultimate karaoke experience!* https://letssing.world/
[4] 2025. *MakeMusic Cloud.* https://www.makemusic.com/makemusic-cloud/
[5] 2023. *MIDI Karaoke.* https://midikaraoke.app/
[6] 2025. *Rock Band.* http://www.harmonixmusic.com/games/rock-band
[7] 2024. *Sync the gameplay with audio and music.* https://docs.godotengine.org/en/stable/tutorials/audio/tutorials/audio/sync_with_audio.html
[8] Fishing Cactus. 2016. *Epistory - Typing Chronicles - Adventure Typing Game.* https://www.epistorygame.com/
[9] Steven M. Demorest. 2001. *Building Choral Excellence: Teaching Sight-singing in the Choral Rehearsal.* Oxford University Press. https://www.google.com/books/edition/Building_Choral_Excellence/D06LmyViPy0C Google-Books-ID: D06LmyViPy0C.
[10] EarMaster. n.d.. *EarMaster - Ear Training.* https://play.google.com/store/apps/details?id=com.earmaster.android&hl=en_US
[11] Behnam Faghih and Joseph Timoney. 2022. Real-time monophonic singing pitch detection. (2022). doi:10.13140/RG.2.2.22054.19526
[12] Virginia Gorlinski. 2025. *SingStar.* https://www.britannica.com/topic/SingStar
[13] Robin Hunicke, Marc Leblanc, and Robert Zubek. 2004. MDA: A Formal Approach to Game Design and Game Research. 1 (2004).
[14] Jane M. Kuehne. 2010. Sight-Singing: Ten Years of Published Research. 29, 1 (2010), 7–14. doi:10.1177/8755123310378453 Publisher: SAGE Publications Inc.
[15] Deeryard LLC. n.d.. *Sight Singing Pro: Solfege.* https://play.google.com/store/apps/details?id=com.deeryard.android.sightsinging&hl=en_US
[16] Gracenotes LLC. 2025. *Practice Sight Reading and Sight Singing Exercises Online.* https://www.sightreadingfactory.com
[17] Lion Theory LLC. 2024. *Eartrainer.io—save hours of grading each week!* https://www.eartrainer.io/
[18] Philip McLeod and Geoff Wyvill. 2005. A Smarter Way to Find pitch. In *International Conference on Mathematics and Computing* (2005). https://api.semanticscholar.org/CorpusID:17432444
[19] Andrew S. Paney. 2015. Singing video games may help improve pitch-matching accuracy. 17, 1 (2015), 48–56. doi:10.1080/14613808.2014.969218 Publisher: Routledge _eprint: https://doi.org/10.1080/14613808.2014.969218.
[20] Nico Schüler. 2021. Modern Approaches to Teaching Sight Singing and Ear Training. 0 (2021), 083–092. https://casopisi.junis.ni.ac.rs/index.php/FUVisArtMus/article/view/7165 Number: 0.