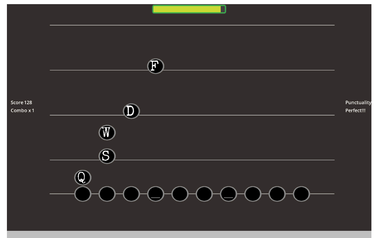
# Syncopated Sentences: A Rhythm-Typing Game

Declan Dury • Dr. Eric Kaltman • COMP 499

## The Game

Syncopated Sentences combines the rhythm and typing genre to create an experience where players type out letters to the beat of a tune. Their ultimate goal is growth in the accuracy and speed of their typing ability while increasing their general familiarity with their keyboard layout. Accompanied by upbeat audio, improving their typing skills should be enjoyable and engaging.




Stage 3:"Final Boss Battle 6"

## Gameplay

The organization of a level is very similar to previous games like *Guitar Hero* and *Dance Dance Revolution*, in which notes fall from above (or below in the case of DDR) towards a target zone on the other end of the screen. However, the number of lanes has increased from a mere four or five to 10 to represent the 10 columns your fingers reside in when abiding by the use of the home row keys. As letters gradually reach the target zone, players are meant to gauge their speed and press the correct key at the correct time to better their score.

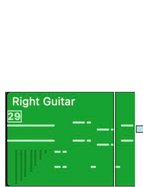
## Development Tools

The game was created using the *Godot* game engine. It has a reputation for being approachable to new game developers and I personally respect the open-source model employed in its development.
Graphical assets were created using the digital painting software, *Krita.*
Midi files were analyzed with Apple's *Garage Band.*

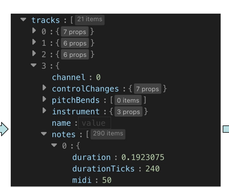


## From Midi Files to Playable Stages

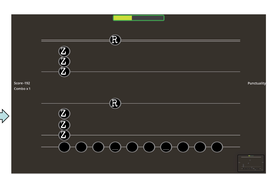
Mapping in-game objects to the tune of a song can be can be a long and arduous process when done manually. So, I made it a goal that I'd find a way to automate this task. I thought midi files would be an excellent candidate for this.
Unlike digital audio formats like mp3, wav, ogg, etc. which record digital sound waves, midi files contain information about the notes in a song such as their pitch, time, and duration. Using a midi-to-JSON conversion tool found as a part of *Tone.js*, I was able to obtain the information necessary for creating a playable map in a format that was easy to parse.
In an effort to make maps/levels as intuitive as possible, I thought it sensible to associate the pitch of each note with a character based on its horizontal position on the keyboard. So if you were to unfurl the keys from the bottom-left to the top-right (z, a, q, x, s, w, etc.), the lowest pitch would be represented with 'z' and the highest with 'p'.
So now that I had a road map for a song in the form of a JSON file, I just had to set a timer at the beginning of the level and when it came time for a note to appear, I checked its pitch, gave it a letter, and sent it down its appropriate column. This method was much more efficient than manually mapping out timestamps and letters.




MIDI

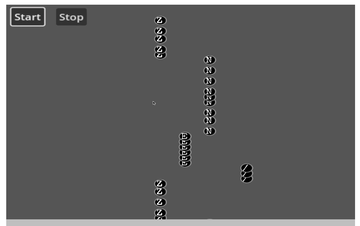



JSON

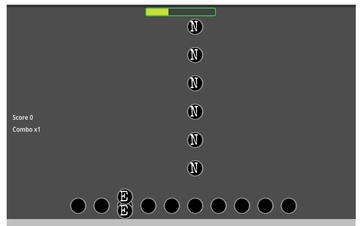


Playable Stage

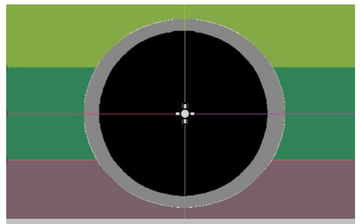
## Progress




1) First instance of choreographed letters

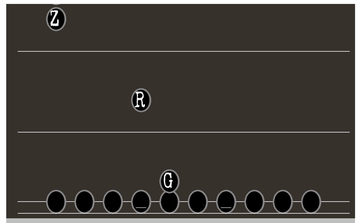



2) Basic Level Layout

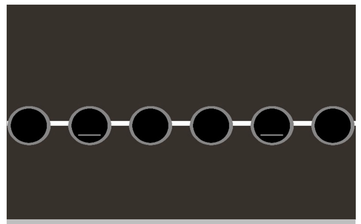


3) Punctuality Zones

4) Added bar lines to facilitate level "readability"

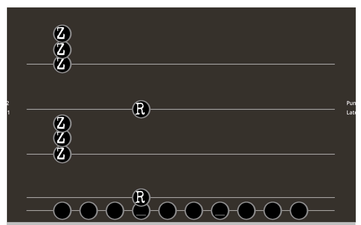


5) Reduced bar line size

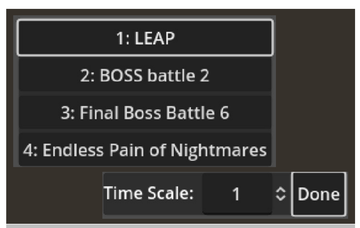


6) Markers on the 'f' and 'j' targets

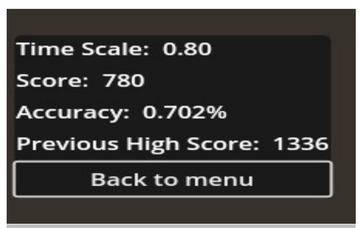

7) End of level statistics



8) Bar lines and letters are in sync




9) Stage Select and Time Scale UI's




10) High scores associated with chosen time scale

## Development

When starting this project, I wanted to see if using midi files to coordinate in-game events was feasible as soon as possible. So, I began hunting down potential midi song candidates.

While browsing OpenGameArt.org I stumbled on the artist "nene". What stuck out most to me was the fact that they included not only the midi file of their song but an associated, higher-quality audio file of the same song. This way, I could benefit from both the parsability of the midi file and the better audio quality of the wav file.

With midi files in hand, I needed a way to bring that note information into the game. Enter the Tone.js Web Audio Framework. This project contained a webpage that converted midi files in JSON files.

JSON files are text files used for storing data. In our case: lists of notes in which each note contains attributes such as pitch and the time it occurs during the song. These files are designed to be easy to read, programmatically, and are perfect as a bridge to bring data from midi files into the game.

Now that I could read the data of each note I can set a timer, and when a note's time occurs, assign a letter and column based on the note's pitch, then send it down the screen (Fig. 1).

Next, I created a basic layout for a level (Fig. 2). This consisted of ten targets at the bottom of the screen which are meant to correlate with the layout of a typical US keyboard. This was to improve/utilize familiarity with the layout and encourage use of the home row keys. A score counter and progress bar were also added.

Rather than simply having a hit or a miss contribute to the player's score, I added a punctuality system (Fig. 3). Players are awarded varying degrees of points based on whether they were early, late, or perfect.

The game is now playable, but it is difficult to read a chaotic stream of letters and was generally overwhelming when tested. So, several changes were made communicate timing and provide feedback on entered keys. First was the addition of bar lines (Fig. 4). These are meant to occur on the strong beats of a song (what you would typically tap your foot or bob your head to). Initially stylized, the first bar lines were too distracting and didn't provide enough fidelity in determining the position of the beat; thus, they were simplified. This way, players could better predict when an upcoming letter was supposed to be pressed by gauging its position relative to the bar lines. Although, the bar lines and letters were not in sync initially and took some time to properly align them.

Targets were made to change color depending on a keystroke's accuracy and markers were added to the 'f' and 'j' targets (Fig. 6) to reduce friction in determining what fingers should be responsible for upcoming letters.

A notification of the player's score and accuracy was added to the end of a level to inform them of their progress and motivate them to improve (Fig. 7).

The bar line and letter discrepancy were finally mitigated (Fig. 8). Even though the time at which a new letter or bar should spawn was checked about every 17ms (0.017 seconds) a discrepancy between the two always seemed to plague me and still does (albeit, nowhere near as noticeable or frequent). I found a solution I was content with; however, it is not comprehensive.

Finally, a UI was added to allow the user to select a level as well as set the time scale for the level they were going to play (Fig. 9). The time scale feature allows users to adjust the speed of the song from 0.1x to 2x. So, when they are first learning a map, the speed can be adjusted as desired and incremented as they get better. The player's high scores are also logged separately for each time scale (Fig. 10).

## Conclusions

My goals for this project were for it to be fun and engaging while reinforcing good typing habits. In my personal opinion and experience I think I have achieved this. I enjoy myself when playing through a stage and have noticed an improvement in my typing ability in recent months.

## Acknowledgements

I'm very grateful to my advisor, Dr. Eric Kaltman for his mentorship, tutelage, and shared enthusiasm for this medium. Through which, I have gained a new set of tools to analyze and appreciate games and software to a degree I previously was not capable of.

## Attributions

Music for *Syncopated Sentences* was provided by artist "nene" via OpenGameArt.org under the creative commons license.
https://opengameart.org/users/nene