USER MANUAL

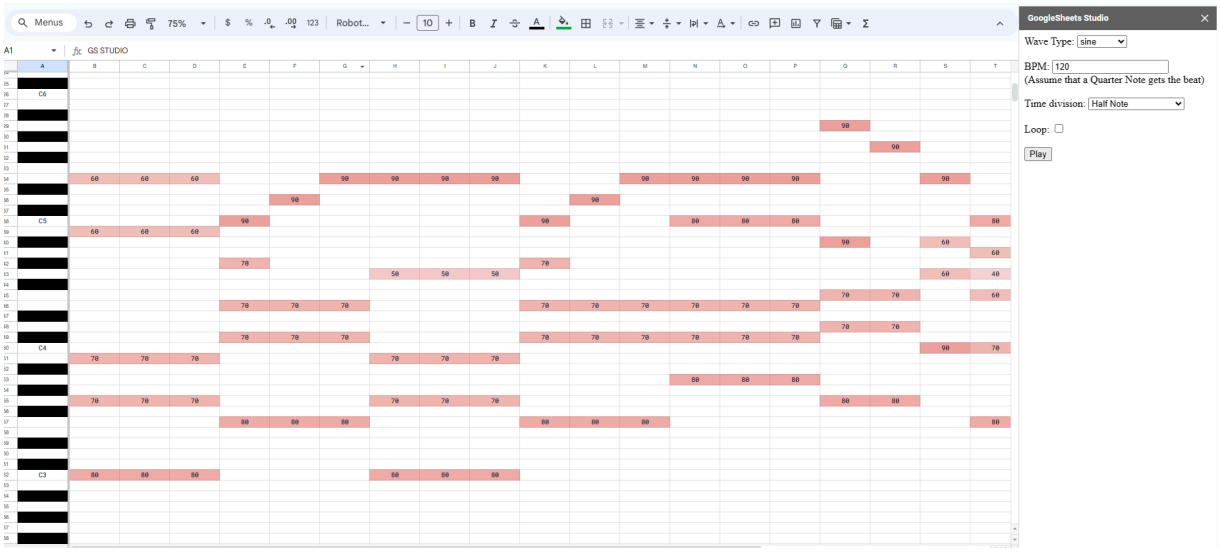
Google Sheets Studios

Using Google Sheets as a Piano Roll

Antoine Nguyen
antoinbn@uci.edu
Computer Science and Engineering, BS — UCI 2024
20wontons.github.io

Abstract

Google Sheets Studio is a Piano Roll in Google Sheets using Google App Scripts and WebAudioAPI. From the data presented in the file, a sequence of notes and pitches will be played from the Add-on. Reading the rows as musical pitches — low-numbered as low notes, and high-numbered as high notes — and the columns as musical time — left-to-right, with each column as a specified note/beat duration — then the spreadsheet can act as a quantized piano roll.

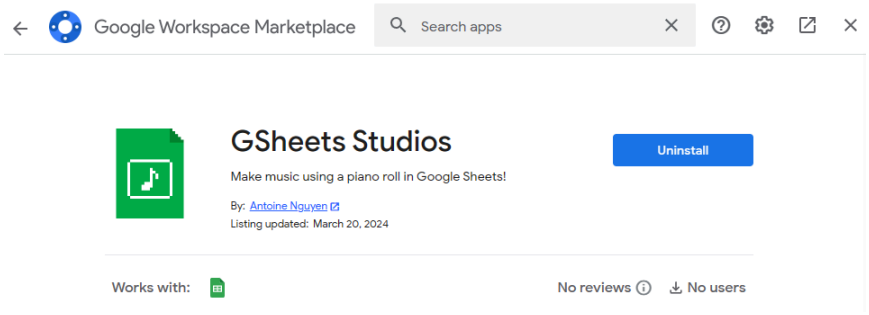


Installation / Setup

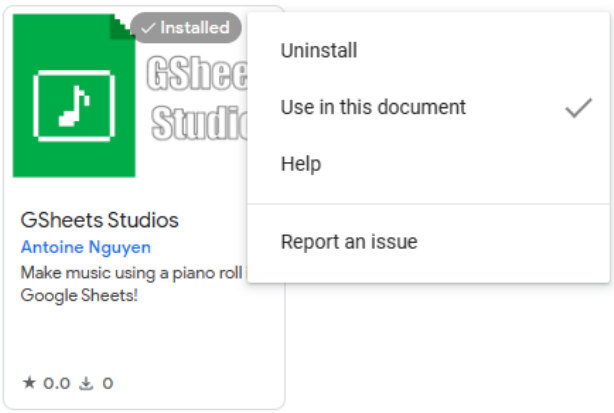
Google Workspace Marketplace

https://workspace.google.com/marketplace/app/gsheets_studios/368428396914

- You will need to use your UCI email to login and access the extension. Click "Install" and allow permissions. The extension does not collect any information and just reads from the spreadsheet to calculate note values.



- After installing, open up a new spreadsheet and go to "Extensions" > "GSheets Studios" > "Start"
- CAUTION: Try the extension on a NEW spreadsheet, as starting the extension will reformat the spreadsheet and information may be lost.
- TROUBLESHOOTING: After installing, the "Start" option may not appear for a spreadsheet. Go to "Extensions" > "Add-ons" > "Manage Add-ons" > Find GSheets Studios > "Use in this document". This will enable the add-on for the spreadsheet and the "Start" option will appear.



Manual Setup

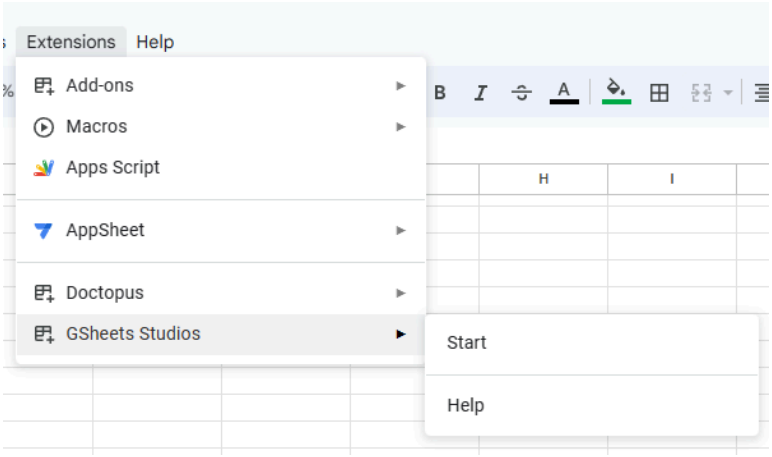
If the marketplace link doesn't work, here are the steps to manually set it up:

1. Go to https://script.google.com/home/ and create a "New project"
2. In "Project Settings", uncheck the option "Enable Chrome V8 runtime"
3. Download the code files from the CAMP folder or from the Github repo
4. In "Editor" add all the code files

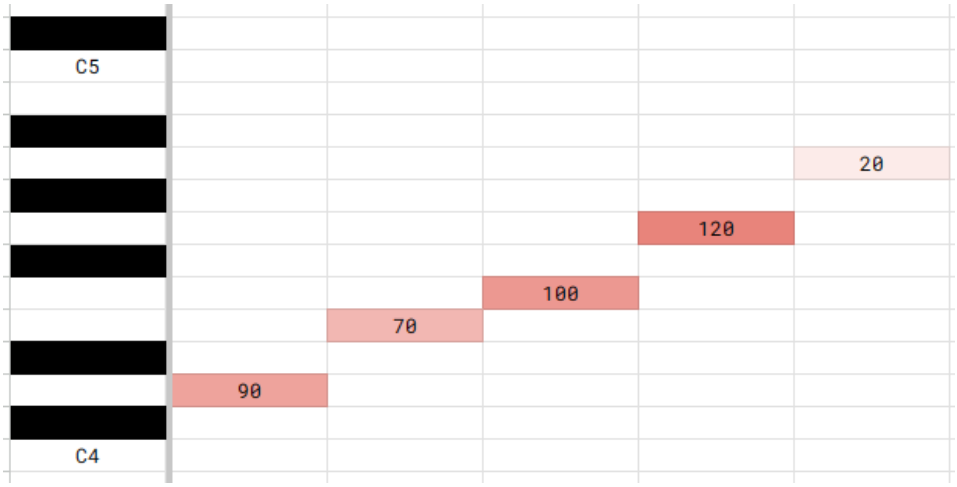
5. To test, navigate "Deploy" > "Test Deployments" and click on "Add test". For the "Config" setting, choose "Installed and Enabled". For the "Test Document", create a new or choose an existing Google Sheets file. Click "Save Test". Select the radio button for that test deployment and click "Execute".
6. After the test file opens and loads fully, navigate "Extensions" > "DAW" > "Show sidebar". Add velocity values to the spreadsheet as detailed in the project description and click "Play" to hear the audio.

Usage

1. After setting up, open up a new spreadsheet and go to "Extensions" > "GSheets Studios" (or "Google Sheets Studios") > "Start". This will format the spreadsheet with a piano roll in the first column and a time bar in the first row. This will also set up conditional formatting for any velocity values in the field.

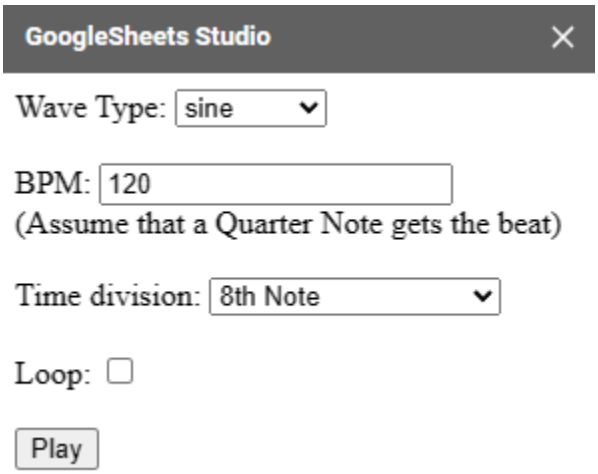


2. Making music: In the field, add velocity values 0-127. Any values outside of that range will be clipped to 0 or 127. Velocities will be colored based on value, ie. higher values will be darker red, and lower values will be lighter red.



3. The sidebar features some options:
    a. Wave type: sine (default), square, sawtooth, triangle. This will change the oscillator type for a different timbre.
    b. BPM: integer value (default: 120). This will change the beats per minute, with the fixed assumption that a quarter note has the beat (think 4/4 time!).

c. Time-division: Whole Note, Half Note, Half Note Triplets, Quarter Note, Quarter Note Triplets, 8th Note (default), 8th Note Triplets, 16th Note, 16th Note Triplets, 32nd Note, 32nd Note Triplets. This will set the time division for each column, that is, what note duration value each column represents.
d. Loop: Toggle on / off (default). Toggling this on will loop the current sequence.
e. Play: Start the sequence audio. Note that when a sequence is playing, there is no current option to pause or stop until the entire sequence finishes. To end a looping sequence, toggle off loop.



Try out my test document with some music samples:
https://docs.google.com/spreadsheets/d/1Vh6-B2BkT4nmmGa0BthsW_A35z-fsN5i52cUhQnU7zc/

- Fur Elise: demonstrates notes and velocity values.
- Sunny Road: demonstrates chords.

Future Considerations

For the sake of time, these ideas were not able to be implemented.

- Holding out a note: Use an asterisk symbol (*) in the cell to the right of a velocity value to hold out that note and velocity.
    - Code Idea: for every row j non-empty cell in a column i — sequence[i][j] — check if the cell in the next column has an asterisk — sequence[i+1][j]. If so, then add the oscillator, amp, and amplitude to an array of persisting notes. Check that array for every column after the first, and allow the note to continue, that is, do not stop the oscillator, until there is no asterisk detected in the next cell.
- Changing the tone of notes:
    - Currently, there is a dropdown menu for choosing the wave type — sine, square, sawtooth, and triangle. These are all oscillators.
    - I can look into Tone.js to use synth sounds and instrument samples.
- Reading from multiple sheets — instrument tracks.

References

- Creating an Add-on using Google Apps Script
- Apps Scripts HTML Service: Best Practices
- Motivation: https://youtu.be/To2JIXGoYzA and https://youtu.be/RFdCM2kHL64

Using WebAudioAPI (or any JS library) in Apps Scripts

Google App Scripts uses a JavaScript-based language called Google Scripts (.gs), but it presents different capabilities than JavaScript. Google Script is used to read, write, and manipulate data from Google Sheets (or really any Google App you want). However, it is unable to use WebAudioAPI or any API in general since it is specifically for Google Apps.

The workaround I was able to find organically, however, was that you are allowed to add HTML files to your Google App Script, originally to be used to create a sidebar. In the HTML file, you can add a <script> tag and use JavaScript with all its functionalities as usual (including WebAudioAPI!).

This was affirmed when I found a "best practices" article from the Google Developers website that encouraged separating the HTML, CSS, and JavaScript into their individual .html files. That article shows a "Javascript.html" file that begins with the <script> tag and ends with the </script> tag, and all the JavaScript in between. Organizing the JS this way allows me to neatly use WebAudioAPI in my project.