Les Paulverizer Website

Casey Costa, Lexi Krzywicki, Tristin Youtz

Project Summary

The Les Paulverizer is a small USB MIDI device that allows a user to play looping background tracks while they play melodies on their primary instrument. In previous iterations of this project, this functionality was provided through a Max 8 patch. During A term 2024, we were tasked with emulating the functions of this Max patch within a web platform. The website needed to connect to a MIDI device to trigger four buttons that could start and stop quantized audio loops. We used the ReactJS library to build the website and GitHub for source control. Below is our Gantt Chart of project objectives and our expected completion date for each. This chart was followed very closely throughout the term.

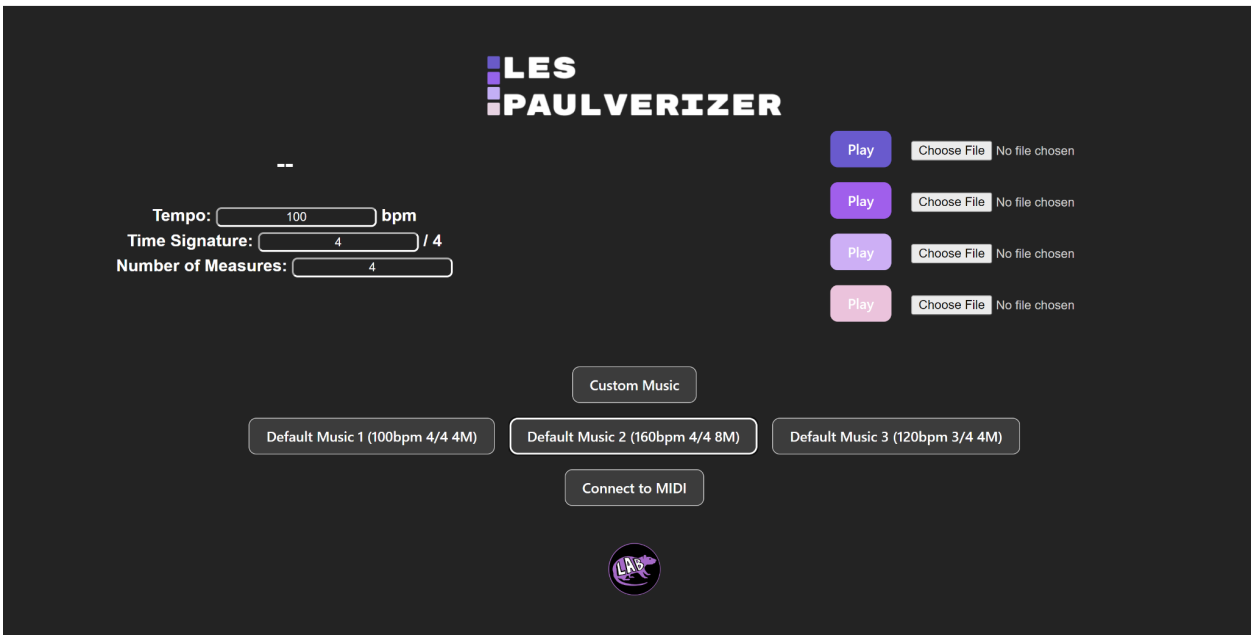
	Week Number						
Objective	**1**	**2**	**3**	**4**	**5**	**6**	**7**
Define project goals	X						
Establish website framework	X						
Establish Github repository	X						
Create website buttons that play sounds		X	X				
Create means of importing WAV/MP3 files		X	X				
Create sound looping functionality		X	X	X			
Add start and stop functionality			X	X			
Add ability to adjust tempo for looping			X	X	X		
Interface with USB device			X	X	X	X	
Assign Les Paulerizer buttons to audio files				X	X		
Make website aesthetic					X	X	X
Create summative presentation/demo						X	X

Accomplishments

During the course of this project we:

- Established the website framework using ReactJS and Vite
- Created a GitHub Repository for source control
- Created functionality for the website to import and play audio files
- Created four individual buttons that each had separate audio files
- Implemented a metronome with an adjustable tempo, time signature, and number of measures
- Included functionality that queues audio files to be played on the next downbeat of the following measure
- Allowed audio files to loop nearly gaplessly (fully lossless looping is difficult and out of the scope of this project)
- Interface a MIDI device with the website
- Interact with the website using the physical Les Paulverizer
- Created three original tracks to use as default music for the website
- Adjusted the website aesthetic to match the Les Paulverizer and be aesthetically pleasing

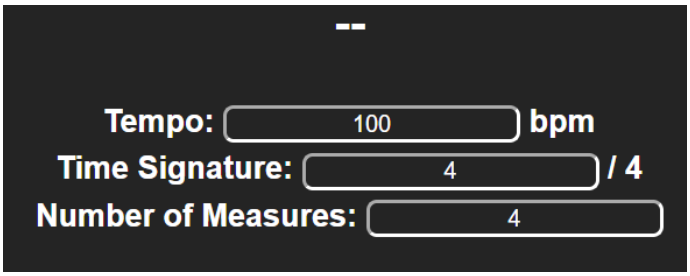
The final website interface is shown below:



Key Functionality

Metronome

The timing functionality of this website depends on a **global metronome**, displayed at the top of the website.



The metronome's default state is '--', and it waits for the user to trigger any of the audio files. When the user triggers any audio files, the metronome turns on and begins counting. For example, at 60 bpm, 4/4, and 4 measures, the metronome count from 1-16, with an interval time of 1 second. After reaching 16, the counter will be set back to 1. This repeats until all buttons are turned off, resetting the metronome to the default state.
When active, the metronome uses the following procedure:

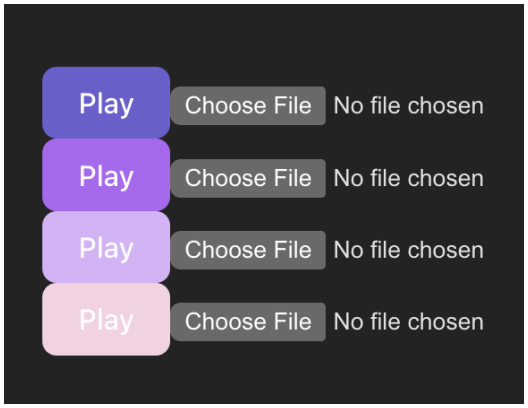
1. Start at 1
2. Count by regular intervals (set by the **Tempo** text box)
3. Count up to the total number of beats in the given measures (number of beats per measure multiplied by number of measures, set by **Time Signature** and **Number of Measures**)
4. Loop back to 1 and repeat

Note: Interacting with any of the input text boxes will stop and reset all the audio.

Sound Buttons

We implemented **four toggleable buttons** in the center of the website, mirroring the four buttons on the Les Paulverizer. Users can click on the buttons directly or connect the Les Paulverizer to send MIDI signals that toggle the buttons. The four buttons are activated using the following MIDI ID's: 63, 65, 67, and 69 (corresponding to the notes D#5, F5, G5, and A5). Users can connect any MIDI device which can play these notes, and it will interact with the buttons on the website.

The buttons switch between **Play** and **Stop**. In "Play" mode, the website doesn't play any audio for that track. In "Stop" mode, the website plays audio for that track. See below:

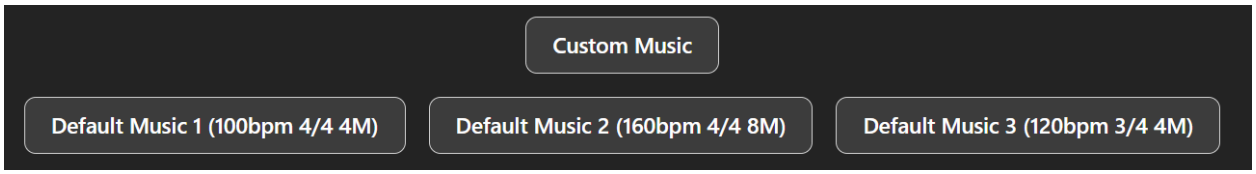


When the user clicks “Play” on any of the buttons, two things may happen. First, if the **global metronome** is off, then clicking on a “Play” button will trigger the metronome to start, and the audio will play. Second, if the global metronome is on, the sound will wait to play until the metronome reaches beat 1.

When the user clicks “Stop” on any of the buttons, that audio file will immediately stop playing. When the metronome reaches beat 1, the website will loop each audio file currently playing. Thus, users can keep tracks playing at a quantized time interval.

Sound Selection System

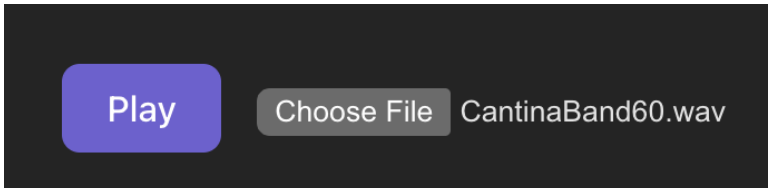
As part of the Les Paulverizer site, we created **3 default presets** to experiment with and the option to upload your own music samples.



The presets each have default values for the tempo, time signature, and number of measures, noted on each of the buttons. Each preset is made by a member of our group:

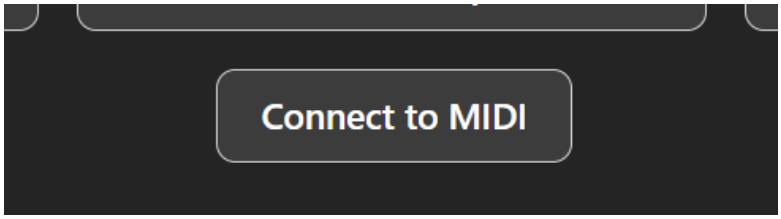
- Default Music 1: Lexi Krzywicki
- Default Music 2: Tristin Youtz
- Default Music 3: Casey Costa

The button on the top, **“Custom Music,”** allows the user to play their own uploaded music samples. Music can be uploaded directly to the right of the corresponding buttons. The user can drag-and-drop samples (.mp3 or .wav) into the interface or click on the “Choose File” button to open a file browser. Without uploading any audio files, switching to “Custom Music” will play samples from “Default Music 1.”

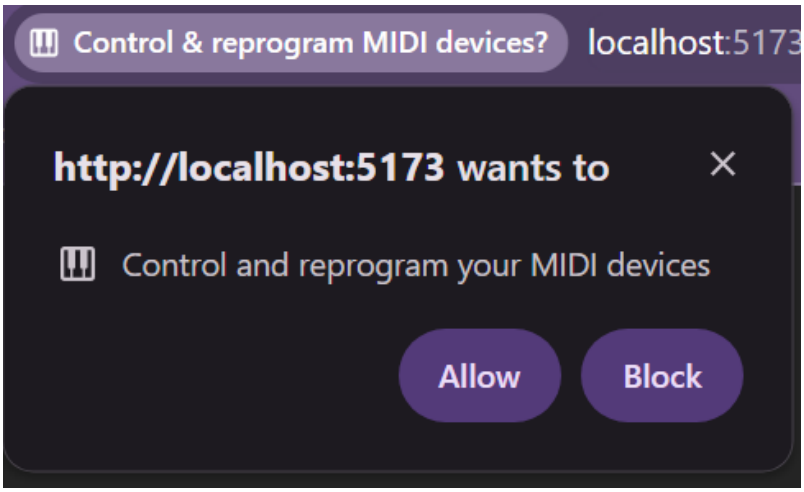


Connecting with a MIDI Device

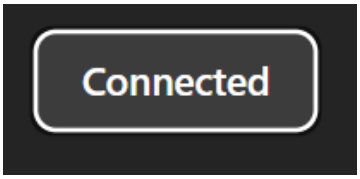
The final piece of functionality on the website is the **Connect to MIDI** button.



When pressed, the website will prompt the user for permission to connect to MIDI devices. This looks different for each web browser, but an example of this message from Google Chrome is shown below:



When the user presses "Allow" the website will scan the computer for MIDI devices and hail the first device found. The status of this process will be displayed on the button itself:



When the button reads "**Connected"**, the website is ready for MIDI input. This was achieved using the Web MIDI API, which returns a given MIDI message as an integer array with the following information:

- **Message Type** (ex. 144 for note onset)
- **Note Pitch** (integer value for all notes from octaves 0-10)

- **Note Velocity** (volume expressed in a range from 0-127)

As previously described, the Les Paulverizer module is mapped to notes 63, 65, 67, and 69, so mapping the onset of these four notes to the website buttons can allow the MIDI device to access the full website functionality.

High-Level Code Logic

We built this website using the React framework, which is based around "states" and "listeners." These are tied to rendering and re-rendering elements on the website; for instance, when an object's "state" is changed from one value or another, the website will rerender to display that change, and other functions can listen for that value change to run calculations.

For React, these are conducted through the **useState** and **useEffect** hooks. The useState states are used for tracking values such as:

- Metronome value
- User input selections (tempo, time signature, number of measures)
- Object statuses (button names, metronome state)

Conversely, there are two primary useEffect Listeners that drive the website are the **Timer Handler** and the **Sound Handler**.

The **Timer Handler** is a useEffect-based function that listens for changes in the **metronome status**. This status is governed by the logic for the buttons to exhibit the behavior described previously. When the value is set to "true", the Timer Handler runs the setInterval hook to begin counting the metronome up at the defined intervals. Otherwise, the interval is cleared and the metronome value is reset.

The **Sound Handler** is a useEffect-based function that listens for changes in the **metronome value.** As previously described, the audio loops are quantized by all playing on the given downbeat. Thus, whenever the listener reads a value change, it will check to see if the value is Beat 1. If this is the case, the listener will reset and play all loops whose buttons are set to the playing state.

The code itself has a much more in-depth explanation of this functionality and the code that governs the rest of the website. This can be found here.

Next Steps

Currently, the Les Paulverizer device **only works on macOS**. In the Arduino code provided by previous projects, the Les Paulverizer module appears to attempt a combination of Bluetooth Low Energy (BLE) communication via the **ArduinoBLE library** and traditional serial communication via the **MidiUSB** library. In practice, the BLE protocol was never proven to be successful, and MacOS was the only device that could assign the Arduino as a MIDI device to read. Fixing this will require troubleshooting the Arduino code or attempting a new communication method altogether.

One way to make this project compatible with windows would be to use **raw serial communication** instead of MIDI. This would remove the need for custom device assignment, which seemed to be the issue with Windows. The team explored this briefly but did not have the time to fully integrate this solution. The Vite distribution also notably caused several issues with this integration attempt, as well as limiting our ability to use the console log feature for debugging.

Resources

- Project Info:
    - https://vjmedia.wpi.edu/Private:Project_Personnel
    - https://vjmedia.wpi.edu/Private:Les_Paulverizer
- ReactJS
    - React JS: https://react.dev/
    - Learning React: https://react.dev/learn
    - Functions to run:
        - npm run dev
- Vite
- GitHub
    - Repository: https://github.com/ElectricGuitarInnovationLab/LesPaulverizer
- Physical Les Paulverizer
- Connecting Arduino to WiFi via USB:
    - https://www.youtube.com/watch?v=emNxwWpuGvI
- HTML Audio Constructor
    - HTMLAudioElement - Web APIs | MDN
- Stack Overflow
    - React JS upload audio from local machine and play in browser - Stack Overflow
    - How to Setup a setInterval Timer Properly in a React Functional Component? - Stack Overflow
    - Playing consecutive audio sounds in ReactJS. Delay issue when live - Stack Overflow
- W3 Schools
    - React useEffect Hooks
    - HTML Audio/Video DOM Reference
- Geeks for Geeks
    - https://www.geeksforgeeks.org/create-a-stop-watch-using-reactjs/
- Max 8 Download: https://cycling74.com/downloads
- Max RNBO: https://rnbo.cycling74.com/
- Hooks
    - https://www.npmjs.com/package/react-drag-drop-files
    - 6 Ways to Stretch a Background Image with CSS | Cloudinary