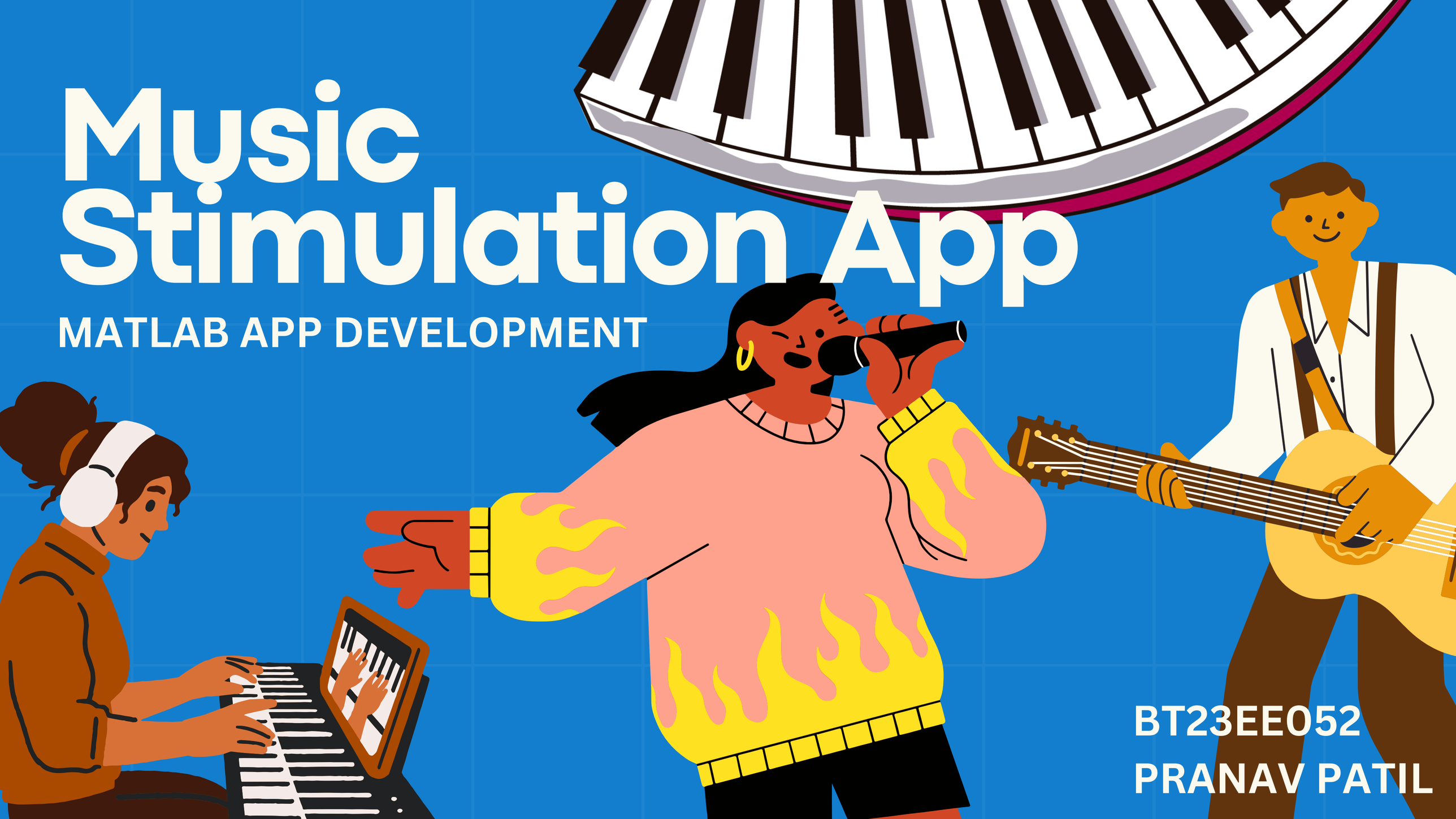
Music Stimulation App

MATLAB APP DEVELOPMENT

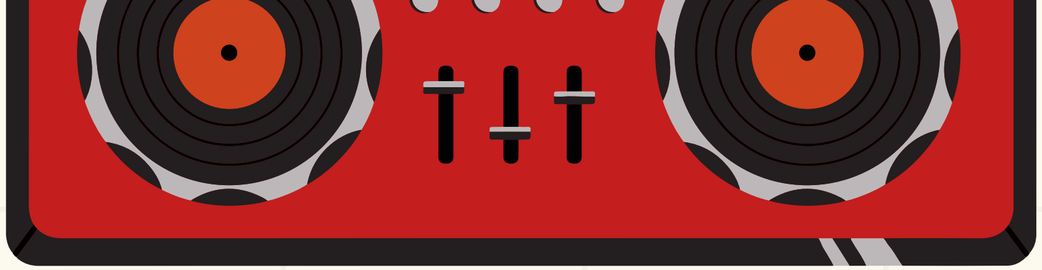


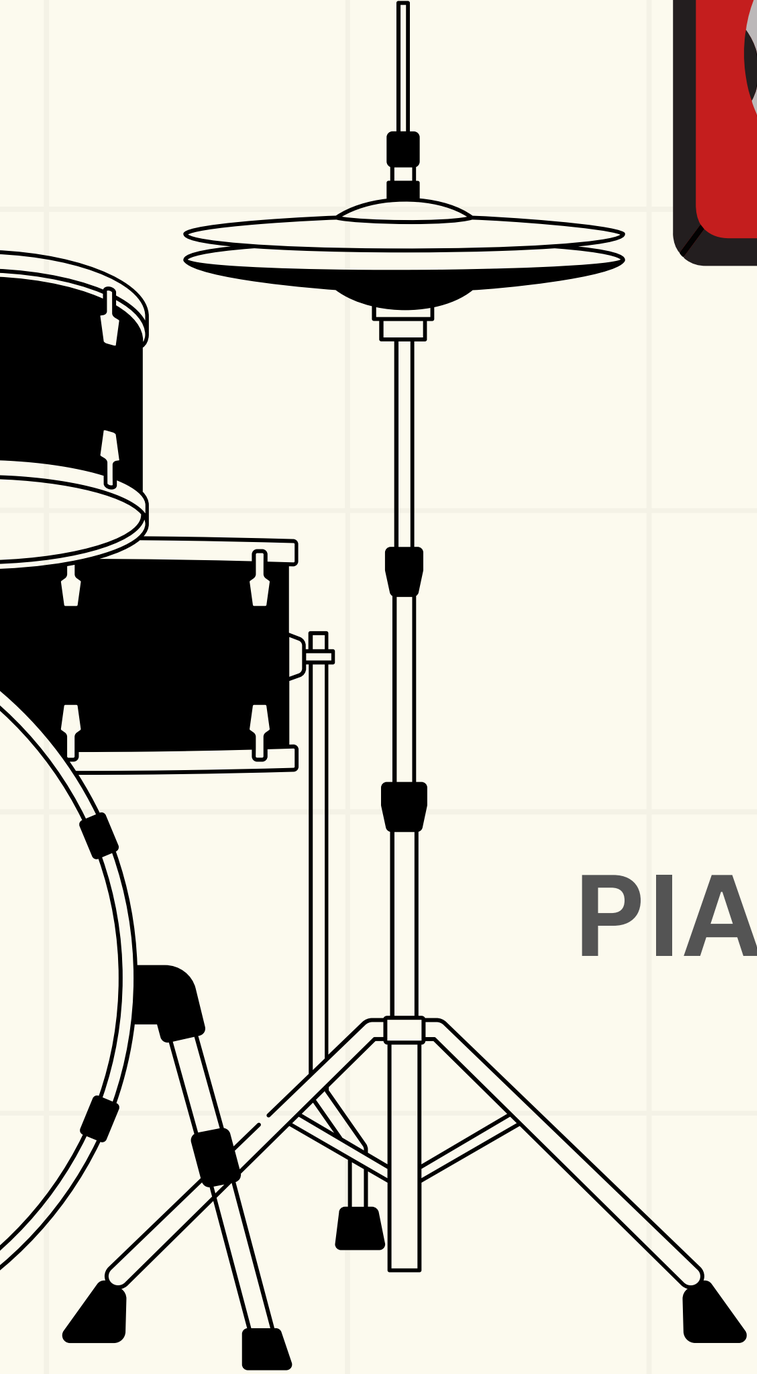
BT23EE052
PRANAV PATIL

Overview

MATLAB App for Music Simulation allows users to create, analyze, and modify sounds using an interactive application. It helps in generating musical tones, processing audio signals, and experimenting with different sound instuments , it virtually facilates the feeling of playing instrument and is accessible anywhere if you carrying your device with you , also beneficial as it helps to have idea about instrument without purchasing expensive ones .

MATLAB provides built-in functions for handling sound, making it easy to load, process, and play audio files. This is useful for musicians, researchers, and students interested in digital signal processing. With a user-friendly interface, even beginners can explore and experiment with music and sound simulations.





Elements of App

PIANO , DRUMSET , OCTAPAD

- DESIGNING
- CALLBACKS
- APP WINDOW

DESIGNING

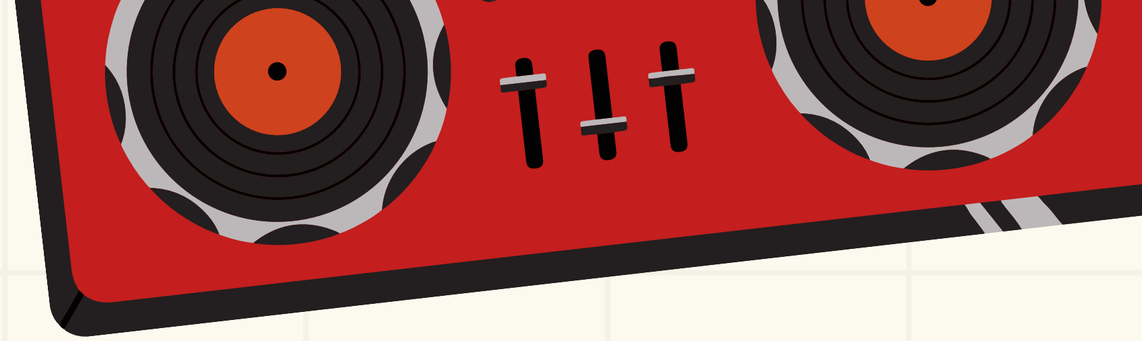
1) Buttons have been placed and arranged as piano keys giving them black and white colors

2) A wooden template is used as image and ‘send backward’ option is used so all buttons come on top layer

3)buttons with song names are placed at top portion.

4)RECORD ON and RECORD OFF buttons are made by placing images in icons

DESIGNING

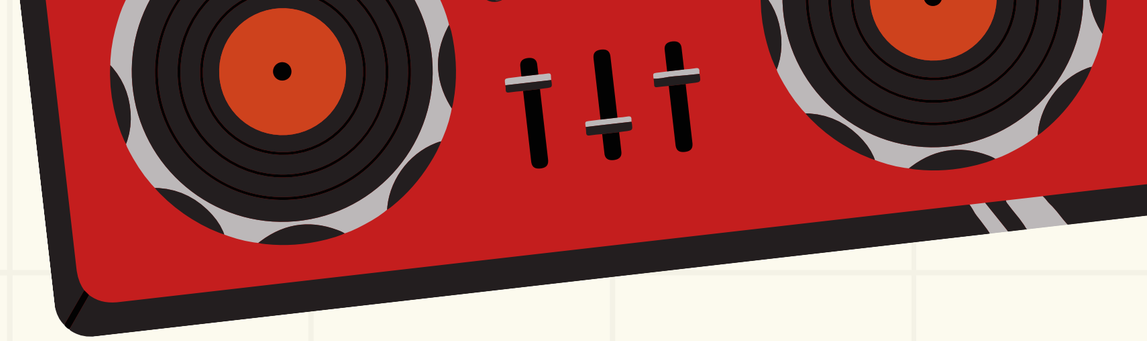


1)Drum image is downloaded , button icon images are cropped for specific elements.

2)Drum image is placed and is ‘send backward’. now buttons are placed on each elements and are resized according to cropped icon images. and image icons are inserted.

3)Record buttons are placed

DESIGNING



1)Octapad image is downloaded , button icon images are cropped for specific elements.

2)Octapad image is placed and is ‘sent backward’. now buttons are placed on each elements and are resized according to cropped icon images. and image icons are inserted.

3)Record buttons are placed

Callbacks

1) Producing Sound in Piano using Piano Theory

In music theory there is a order of notes called CHROMATIC ORDER which comes in sequence because the frequency increase in steps , these steps are called SEMITONES

C, C#, D, D#, E, F, F#, G, G#, A, A#, B

eg : C to C# jump is one semitone jump.

$$f = 440 * 2^{(n/12)}$$

this piano frequency formula can be used to produce any musical note , it has reference 440Hz that is A4 note , n is the no. of semitone jump can be positive or negative depending upon the desired note has more frequency or less frequency than notE A4

Callbacks

EXAMPLE : producing C5 note

```
function C5ButtonPushed(app, event)

    fs = 44100;          % Sampling frequency
    n = 3;               % C5 is 3 semitones above A4
    f = 440 * 2^(n/12);% Formula: f = 440 × 2^(n/12)
    duration = 3;        % Duration in seconds
    t = 0:1/fs:duration;
    y = sin(2 * pi * f * t) .* linspace(1, 0, length(t)); % With fade out
    sound(y, fs);        % Play sound
end
```

In such a way every note on piano can be produced

NOTE : In my app I've used preset audios to play piano notes , this is a way by which all professional pianos are made but here in MATLAB it was producing a sound which was like frequency beeps and were not justifying the musicality of a piano. For that Sampling frequency has to be same with default MATLAB sampling frequency, and it even includes the concept of quality of sound , eg; every singer can sing same note still the quality differs , in same way it produces beep frequencies but its not the sound which have quality of piano

Callbacks

2) Palying preset sound when button is pressed

```
function C5ButtonPushed(app, event)
    [y, Fs] = audioread('c5.mp3');
sound(y, Fs);
```

- audioread reads the audio file 'c5.mp3'.
- y = audio data (samples).
- Fs = sampling frequency of the audio file.
- sound plays the audio data y at sampling rate Fs.

Similarly buttons are assigned the sounds , this is used in paino , drums , oactapad , these preset sound files should be in working MATLAB directory

Callbacks

3) Recording the played music

recording on:

```
        global recObj;
recObj = audiorecorder(44100, 16, 1); % Ensure recObj is an audiorecorder
uialert(app.UIFigure, 'recording on', 'recording...');

record(recObj);
```

- recObj declares as a global variable so it can be accessed from other callbacks (e.g., Stop button).
- Creates an audio recorder object with: 44100 Hz sampling rate (CD Quality) ,16-bit resolution ,1 channel (Mono sound)
- uialert : Displays a pop-up message box on the App window saying "recording on" with title "recording..." as a notification to the user
- record(recObj); → Starts recording audio using the recObj recorder

Callbacks

3) Recording the played music

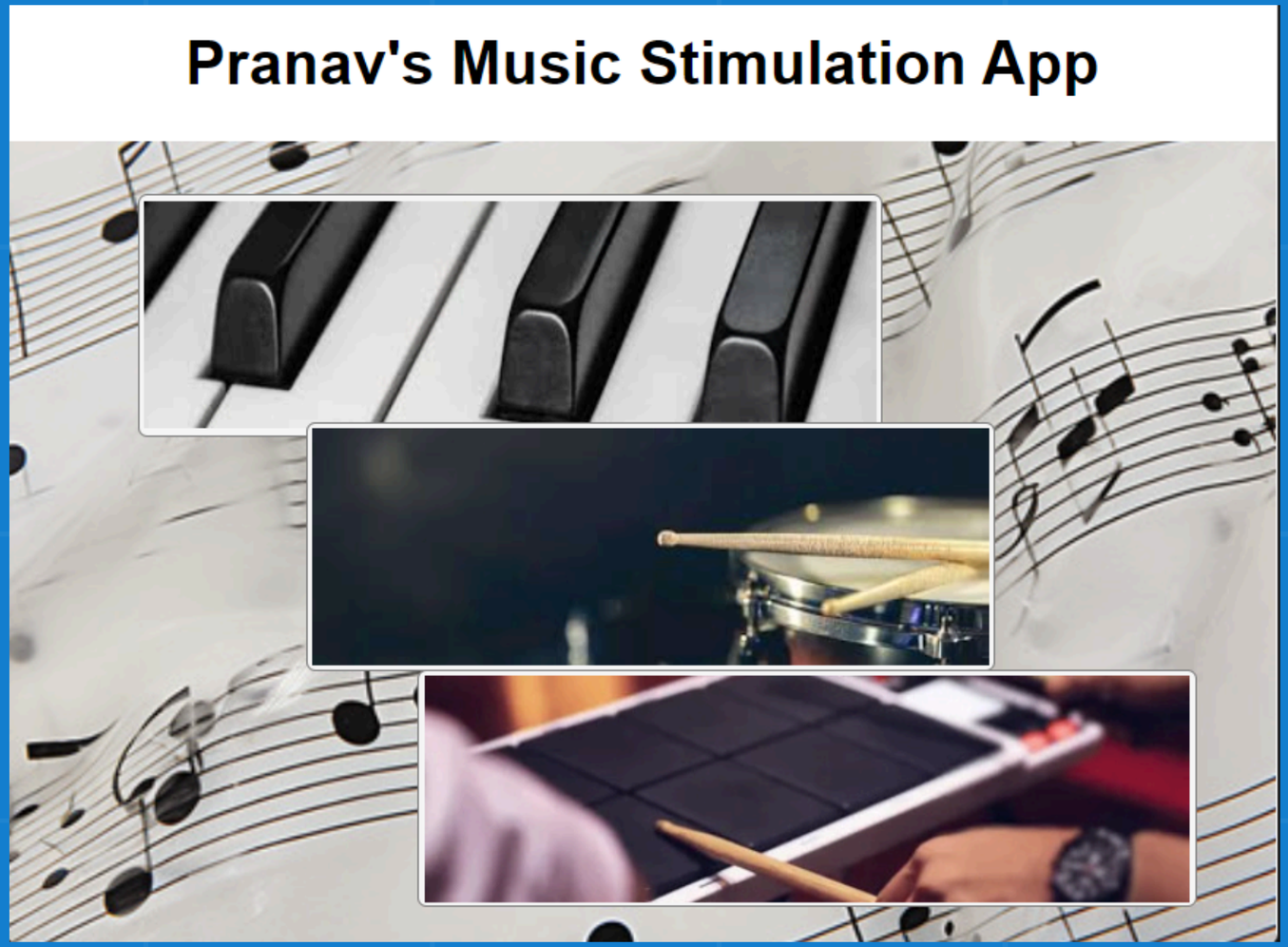
recording off:

```
    function recordoffPushed(app, event)
        global recObj;

uialert(app.UIFigure, 'recording off', 'recording...');
stop(recObj);
audioData = getaudiodata(recObj);
filename = fullfile(pwd, 'z_recorded_audio_drums.wav');
audiowrite(filename, audioData, 44100);
disp(['Audio saved as ', filename]);
    end
end
```

- global recObj; → Accesses the global audio recorder object (recObj) that was created when recording started.
- uialert→ Shows an alert message saying "recording off" to notify the user that recording has stopped.
- stop → Stops the ongoing recording.
- audioData = getaudiodata(recObj); → Retrieves the recorded audio data from recObj and stores it in the variable audioData.
- filename = fullfile(pwd, 'z_recorded_audio_drums.wav'); → Creates a full file path in the current working directory with the name z_recorded_audio_drums.wav.
- audiowrite(filename, audioData, 44100); → Writes the recorded audio data to a WAV file with a sampling rate of 44100 Hz
- disp : Displays a message in MATLAB Command Window confirming the saved file name and location.

The Master App



This is new app which helps in opening the intrument simulators from one single app. Image icon is placed on each button signifying the instrument.

This makes it less complex and easier to open any intrument simulator

Master app callback

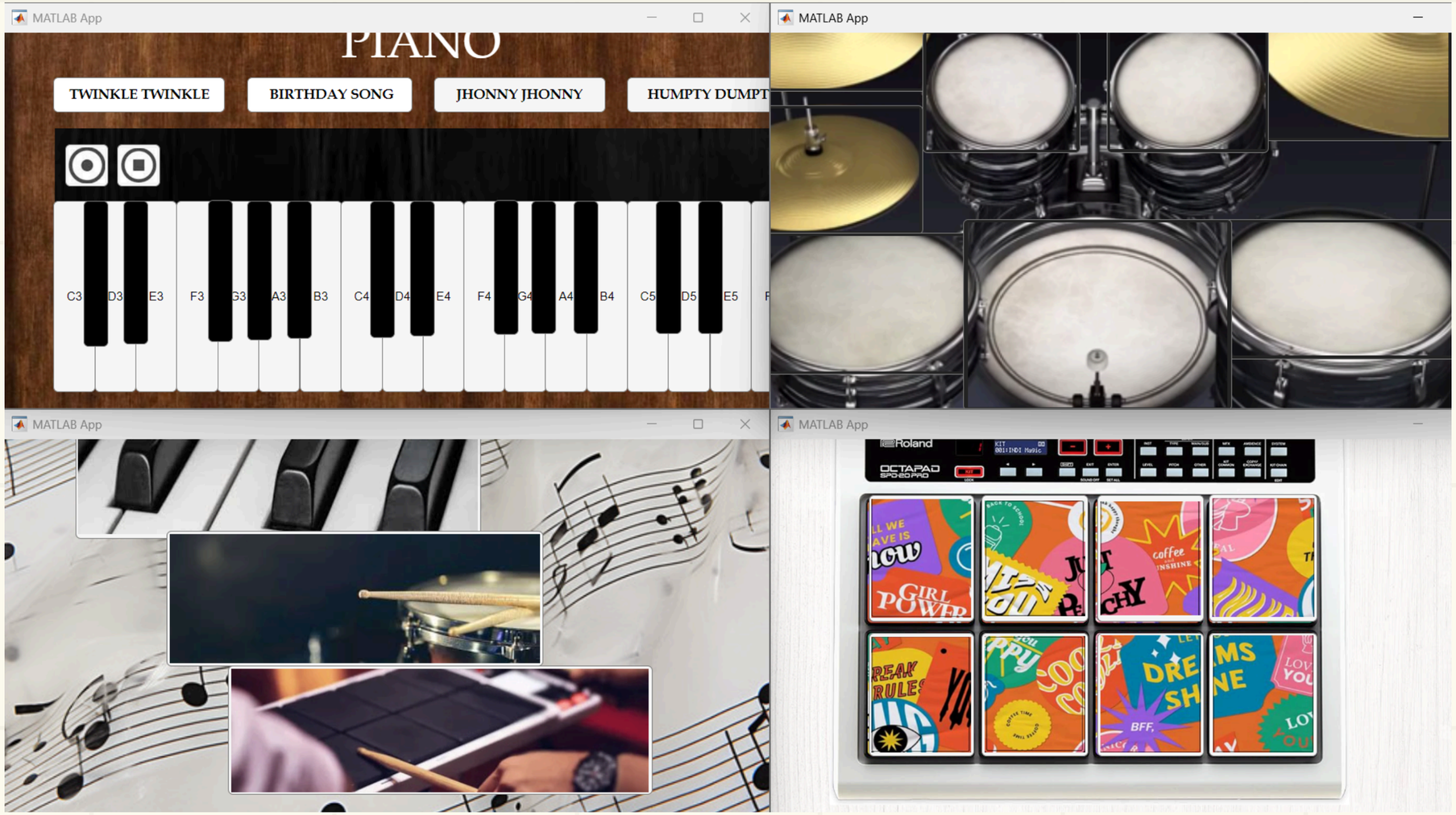
```
function pianoPushed(app, event)
    system('PIANOCRAZY.mlapp &');
```

Executes a system shell command to open another MATLAB App named PIANOCRAZY.mlapp.

- system() → Executes OS-level commands.
- PIANOCRAZY.mlapp → The app you want to launch.
- & → Runs the command in the background (non-blocking).

NOTE : other sub-apps are in working directory

OVERVIEW



ALL APPS OPENED BY THE MASTER APP

Thank You