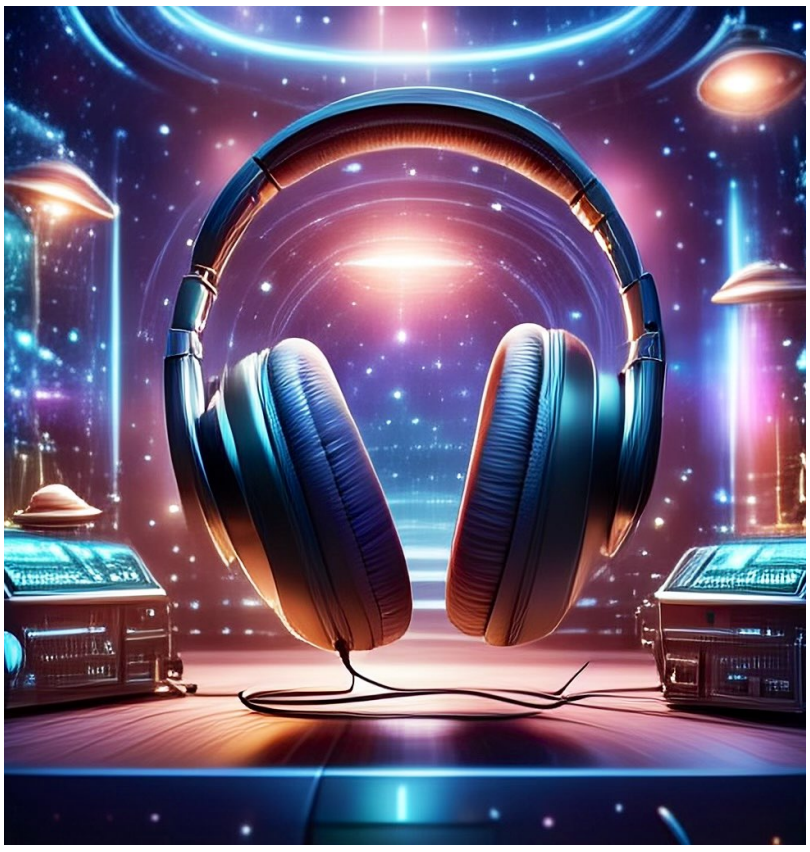
WAVE SMITH

Project on DSP applications using MATLAB

Developed By-
Arshdeep Singh 2310994509
Devraj Singla 2310994520
Gurnoor Singh 2310994525
Jugraj Singh 2310994530
Tanushree Gupta 2310994587



This MATLAB script creates a comprehensive GUI application that integrates Head-Related Transfer Function (HRTF) spatialization with live frequency spectrum (FFT) visualization and dynamic filter controls. It also demonstrates proper handling of DSP states and buffer management.

Note: HRTF stands for *Head-Related Transfer Function*, a method used for audio spatialization. It characterizes how an ear receives a sound from a point in space by accounting for the effects of the head, torso, and pinna. Here, the HRTF data is loaded from a SOFA (Spatially Oriented Format for Acoustics) file.

1. Overview of the Application

- **GUI Layout:** The figure window is divided into two main panels:
  - **Left Panel:** Contains controls for HRTF-based spatialization and a 3D visualization of the sound source.
  - **Right Panel:** Contains a live FFT display, playback controls, and radio buttons to select additional filtering options (e.g., low-pass, band-pass, high-pass), along with an option to set custom filter parameters.
- **Toolkits and Functions:**
  - **GUI Components:** MATLAB's built-in GUI components (e.g., uifigure, uipanel, uicontrol, axes) are used to design the interface.
  - **DSP Objects:** The script utilizes MATLAB's dsp.FIRFilter for applying HRTF filters on the audio stream and audioDeviceWriter to output the processed sound.
  - **SOFA File Handling:** The HRTF measurements are loaded via the sofaread function from a provided SOFA file, allowing the spatial processing to mimic real-world acoustic cues.
  - **Digital Signal Processing:** The script implements FFT for real-time spectral analysis and uses Butterworth filters for additional audio filtering stages.

2. Detailed Code Walkthrough

2.1. Setting Up the GUI

2.1.1. Main Figure Creation

- **Figure Creation:**

```
1. fig = figure('Name','Combined HRTF, Live FFT & Buffer Cleanups','NumberTitle','off', 'Position',[100 100 1280
800],'CloseRequestFcn',@closeFigure)
2.
```

  - A main figure window is instantiated with a fixed size and title.
  - The CloseRequestFcn is linked to the closeFigure callback to ensure proper cleanup upon closing the GUI.

2.1.2. Left Panel – HRTF Controls & 3D Visualization

- **Panel Setup:**

```
1. panelLeft = uipanel('Parent',fig, 'Title','HRTF Controls & 3D Visualization',  'Units','normalized','Position',[0.02 0.02 0.46 0.96]);
2.
```

  - This panel occupies roughly half of the window and is dedicated to spatialization controls.

- **Azimuth & Elevation Sliders:**
  - Two sliders allow the user to control the spatial orientation:
    - **Azimuth Slider:** Ranges from -180° to +180°.
    - **Elevation Slider:** Ranges from -90° to +90°.
  - Each slider updates the corresponding angle labels with real-time values using the updateHRTF callback.

- **Swap Left/Right Checkbox:**

```
1.swapChk=uicontrol(panelLeft,'Style','checkbox','String','SwapL/R','Units','normalized','Position',[0.05 0.68 0.3
0.05],'Value',0,'Callback',@setSwap);
```

  - This checkbox toggles the left-right channel swap during HRTF processing.

- **Audio File Selector Button:**

```
1. uicontrol(panelLeft,'Style','pushbutton', 'String','Select Audio File', 'Units','normalized','Position',[0.05 0.60 0.4 0.07],
'Callback',@selectAudio);
2.
```

  - Opens a file selection dialog to load an audio file. The chosen file is normalized and prepared for playback.

- **3D Visualization Axes:**

```
1. ax3d = axes('Parent',panelLeft,Units','normalized','Position',[0.1 0.05 0.8 0.45]);
2. [sx,sy,sz] = sphere(30);
3. surf(ax3d, sx, sy, sz, 'FaceAlpha',0.1, 'EdgeColor','none');
4. hold(ax3d,'on');
5. hPoint = scatter3(ax3d, 1, 0, 0, 80, 'filled');
```

  - A sphere is drawn to represent the head or the surrounding space.
  - A scatter plot point (hPoint) marks the current source location. This is updated based on the azimuth and elevation slider values.

2.2. Right Panel – Live FFT Display & Filter Controls

2.2.1. FFT Display Setup

- **FFT Axes:**

```
hAxFFT = axes('Parent',panelRight, 'Units','normalized','Position',[0.1 0.45 0.8 0.45]);
(hAxFFT, 'Frequency (Hz)');
ylabel(hAxFFT, 'Magnitude (dB)');
title(hAxFFT, 'Live FFT');
xlim(hAxFFT, [0 fs/2]);
ylim(hAxFFT, [-100 0]);
fftLine = plot(hAxFFT, nan, nan);
```

  - Sets up a plot window that will display a live FFT of the processed audio frames.
  - The FFT uses a Hann window for smooth frequency-domain analysis.
  -

2.2.2. Playback Controls

Play and Pause Buttons:

```
uicontrol(panelRight,'Style','pushbutton','String','Play','Units','normalized','Position',[0.1 0.92 0.35 0.06],'FontSize',12,
'Callback',@playCallback);
uicontrol(panelRight,'Style','pushbutton','String','Pause','Units','normalized','Position',[0.55 0.9 0.35 0.06],FontSize',12,
'Callback',@pauseCallback);
```

  - These buttons start and pause the audio playback, respectively.

2.2.3. Filter Selection and Custom Filter UI

Filter Group (Radio Buttons):

```
filterGroup=uibuttongroup('Parent',panelRight,'Title','FilterSelection','Units','normalized','Position',[0.1 0.10 0.35
0.30],'SelectionChangedFcn',@filterCallback);
```

  - Provides radio buttons for selecting a predefined filter type:
    - None, Low-pass, Band-pass, High-pass.
  - The filterCallback is invoked on change, updating the filter coefficients accordingly using a Butterworth design (via the butter function).

- **Custom Filter Panel:**

```
customPanel=uipanel('Parent',panelRight,'Title','CustomFilter','Units','normalized','Position',[0.55 0.10 0.35 0.16]);
```

- Contains input fields (edit controls) for lower and upper frequency values.
- A button labeled “Set Custom Filter” triggers the customFilterCallback, allowing the user to define a custom band-pass filter.

2.3. Digital Signal Processing (DSP) and State Initialization

2.3.1. Loading HRTF Data

- **SOFA File and HRTF Configuration:**

```
sofaPath = "P0086_Raw_96kHz.sofa";
hrtfData = sofaread(sofaPath);
fs = hrtfData.SamplingRate;
```

- The script loads a SOFA file containing HRTF measurement data using sofaread.
- The sampling frequency (fs) is set to match the HRTF measurement data.

- **Initial FIR Filters:**

```
leftFilt  = dsp.FIRFilter('Numerator', squeeze(hrtfData.Numerator(1,1,:))');
rightFilt = dsp.FIRFilter('Numerator', squeeze(hrtfData.Numerator(1,2,:))');
```

- Two FIR filter objects are created for the left and right ear processing using the first measurement from the HRTF data.
- This sets the baseline spatial characteristics until the user adjusts the azimuth/elevation.

2.3.2. Additional Filter Defaults

- The script initializes additional filtering parameters:

```
currentFilter = 'none';
b = 1; a = 1;
filterState = [];
```

- When no additional filter is selected, the filter coefficients represent a unity (non-altering) filter.
- filterState is used to maintain the memory required for recursive filter implementations.

2.3.3. Playback State Management

- **State Structure:**

```
state.audioData  = [];
state.currentIdx = 1;
state.chunkSize  = 4098;
state.isPlaying  = false;
```

- The state structure holds audio data, playback index, chunk size for audio frame processing, and a flag to control play/pause behavior.

- **Audio Device Writer:**

```
player = audioDeviceWriter('SampleRate',fs,...
  'ChannelMappingSource','Property',...
   'ChannelMapping',[1 2]);
state.player = player;
```

- An audioDeviceWriter object sends the processed audio to the playback device at the defined sampling rate.
- The channel mapping assigns the left and right outputs for stereo playback.

2.3.4. Packing UI and DSP Handles

- **UserData Structure:**

```
fig.UserData = struct(...
 'hrtfData', hrtfData, ...
   'leftFilter', leftFilt, ...
   'rightFilter', rightFilt, ...
   'state', state, ...
   'azSlider', azSlider, ...
  'elSlider', elSlider, ...
   'azLabel', azLabel, ...
   'elLabel', elLabel, ...
    'swapChk', swapChk, ...
  'hPoint', hPoint, ...
   'hAxFFT', hAxFFT, ...
   'fftLine', fftLine, ...
   'currentFilter', currentFilter, ...
   'b', b, 'a', a, ...
   'filterState', filterState, ...
  'lowerEdit', lowerEdit, ...
  'upperEdit', upperEdit);
```

- All relevant handles (UI controls and DSP objects) and state variables are packed into the UserData property of the figure. This makes them easily accessible from all callbacks.

2.4. Main Processing Loop

- **Audio Processing and Visualization Loop:**

```
while ishandle(fig)
   ud = fig.UserData;
   st = ud.state;
```

```
    if st.isPlaying && ~isempty(st.audioData)
        % Determine next chunk boundaries and extract frame.
        % ...
        % Apply additional filter, if enabled.
        % ...
        % HRTF processing using left/right filters.
        % ...
        % Optionally swap channels.
        % ...
        % Send processed audio to device writer.
        % ...
        % Update index, reset buffers for looping.
        % ...
        % Update live FFT display using a Hann window.
        % ...
    end
    drawnow;
    pause(0.001);
end

```

- The loop continuously checks whether the figure is still active.
- If playback is active and audio data is loaded, the script processes the next audio frame:
  - **Chunk Extraction:** A portion of the audio data is selected based on the current index and predefined chunk size.
  - **Additional Filtering:** An optional Butterworth filter is applied if selected; its state is maintained between frames.
  - **HRTF Filtering:** The audio frame is processed with the left and right FIR filters to generate spatial cues.
  - **Channel Swap:** If enabled by the user, the left and right channels are swapped.
  - **Buffer Management:** After playing a chunk, indices and buffers (for custom filters and HRTF) are reset if the end of the file is reached.
  - **Live FFT Update:** The script computes the FFT over the processed frame (using a Hann window), and updates the live spectrum display.

2.5. Callback Functions

2.5.1. updateHRTF

- **Purpose:**
  Updates filter coefficients based on the azimuth and elevation slider values.
- **Steps:**
  - Reads the slider values.
  - Displays the current values next to the sliders.
  - Finds the nearest HRTF measurement by computing a Euclidean distance in the azimuth-elevation space.
  - Updates the FIR filter coefficients for both left and right channels.

- Updates the 3D visualization by computing the new source location coordinates on the unit sphere.

2.5.2. selectAudio

- **Purpose:**
  Prompts the user to select an audio file and processes the file for playback.
- **Steps:**
  - Opens a file dialog and lets the user pick a supported audio file.
  - Converts stereo audio to mono if necessary, resamples it to match the HRTF sampling frequency, and normalizes the amplitude.
  - Resets playback and DSP buffers to prevent any residual artifacts from previous operations.

2.5.3. playCallback and pauseCallback

- **Purpose:**
  Start and pause the playback of the audio file.
- **Steps:**
  - Set the state flag isPlaying accordingly, which the main processing loop reads to determine whether to process and play audio frames.

2.5.4. filterCallback

- **Purpose:**
  Sets additional filtering parameters when the user selects one of the predefined filter options.
- **Steps:**
  - Reads the selected radio button string.
  - Based on the filter type (none, low-pass, band-pass, high-pass), computes the Butterworth filter coefficients using MATLAB's butter function.
  - Resets the filter state so that previous filter outputs do not affect new filter conditions.

2.5.5. customFilterCallback

- **Purpose:**
  Allows the user to set a custom band-pass filter using specific lower and upper frequency limits.
- **Steps:**
  - Reads values from the custom filter text fields.
  - Validates the inputs (ensuring that frequencies are within acceptable range and in correct order).
  - Designs a band-pass Butterworth filter with the user-provided parameters.
  - Resets the filter state to clear any existing filtering artifacts.

2.5.6. setSwap

- **Purpose:**
  Handles changes to the "Swap L/R" checkbox state.
- **Steps:**
  - This function relies on the checkbox's value during processing and triggers an immediate update (using drawnow) if the state changes.

2.5.7. closeFigure

- **Purpose:**
  Ensures proper cleanup when closing the GUI.
- **Steps:**

- Stops audio playback by setting isPlaying to false.
- Releases the audio device writer resource.
- Resets DSP filter objects to clear any remaining internal states.
- Finally, deletes the figure, terminating the application.

3.The Inspiration Behind the Application

Imagine being able to listen to sound the way our ears naturally do—with every nuance, direction, and subtle difference captured just as life intended. This application is born out of that passion: to bridge the gap between theoretical audio processing and the warm, lived experience of real-world sound.

Why do we need it?

- **A Playground for Curiosity:**
  Think of this application as a hands-on laboratory where students, hobbyists, and seasoned researchers alike can experiment with spatial audio. It makes abstract concepts tangible by letting users adjust the perceived sound location in real time, much like an artist mixing colors on a palette.

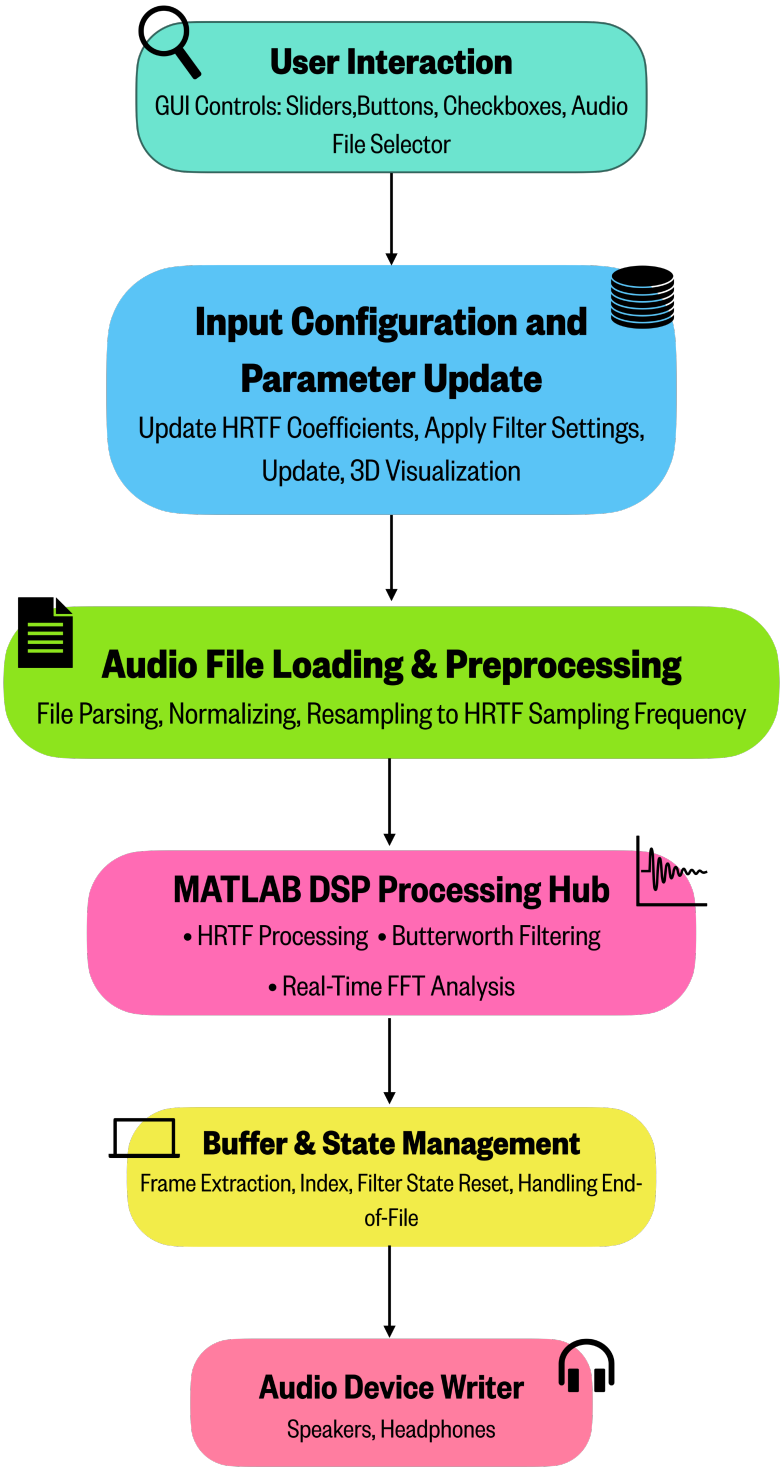
- **Instantaneous Real-Time Magic:**
  The charm of this project lies in its ability to process audio on the fly. As users tweak parameters via intuitive controls, they witness instant changes in the frequency spectrum. This real-time visual and auditory feedback makes it an indispensable tool for anyone eager to dive into the wonders of digital signal processing.

- **Tailored for Diverse Explorations:**
  Whether you're exploring the basics of how sound interacts with the human anatomy or pushing the boundaries with custom filter designs, this application adapts to your creative flow. Its built-in flexibility — offering simple presets along with the power to design your own filters — makes it a versatile companion for your acoustic adventures.

- **An Easy Gateway to Complex Concepts:**
  By integrating visual feedback with interactive DSP routines, the application demystifies sophisticated topics such as HRTF (Head-Related Transfer Function) and FFT (Fast Fourier Transform). It's like having a friendly guide lead you through the symphony of sounds and intricate waves that define our auditory world.

- **Thoughtful Engineering for Seamless Performance:**
  Beyond its creative allure, the design pays homage to the precision required in digital signal processing. Meticulous state management and rigorous buffer handling ensure that even when you're lost in the creative process, everything runs smooth and true—allowing your ideas to flow unhindered.

4.The Flow of Audio Magic: Hardware Interface Block Diagram



User Interaction

- **Components:**
  - **Sliders:** Adjust azimuth and elevation for spatial audio positioning.
  - **Buttons:** Initiate actions (e.g., Play, Pause, Select Audio File).
  - **Checkboxes:** Toggle options like swapping left/right channels.
- **Role:**
  Users provide direct commands and input, which sets the tone for how audio will be manipulated and played. This interactive layer makes the application inviting and user-friendly.

Input Configuration and Parameter Update

- **Components:**
  - **HRTF Parameter Updater:** Reads and applies slider values to update filter coefficients.
  - **Filter Selector:** Adjusts Butterworth filter options based on the selected radio button (none, low-pass, band-pass, high-pass) or custom user input.
  - **3D Visualization:** Updates visual representation of the sound source on a sphere.
- **Role:**
  This stage dynamically configures the processing parameters so that the spatial audio effect reflects the user's real-time adjustments. It sets the stage for personalized audio manipulation.

Audio File Loading & Preprocessing

- **Components:**
  - **File Dialog Selector:** Allows the user to pick an audio file.
  - **Audio Preprocessor:** Normalizes audio, converts stereo to mono, and resamples the data to match the HRTF data's sampling frequency.
- **Role:**
  Ensures the audio input is in the correct format and quality before being processed. This preprocessing step is crucial for maintaining the integrity of the subsequent DSP routines.

MATLAB DSP Processing Hub

- **Sub-Processes:**
  - **HRTF Processing:**
    Uses FIR filters loaded with HRTF data to simulate how sound is spatially modified by the human anatomy.

- **Additional Filtering:**
  Applies optional Butterworth filters (low-, band-, high-pass, or custom) to refine the audio output further.
- **Live FFT Analysis:**
  Computes the FFT in real time to visualize the spectral content of the processed audio, providing immediate feedback on frequency characteristics.

- **Role:**
  This is the core of the application where all the heavy lifting in terms of digital signal processing is performed. It synthesizes the user's inputs with acoustic data to generate the transformed audio signal.

Buffer & State Management

- **Components:**
  - **Frame Extraction:** Manages the sequential division of audio data into chunks for continuous processing.
  - **State Control:** Maintains indices, handles looping or end-of-file conditions, and resets filter states as required.
- **Role:**
  Provides a robust mechanism to handle real-time processing, ensuring that audio frames are processed without interruption and that DSP state is maintained accurately between frames.

Audio Device Writer

- **Components:**
  - **Playback Data Handler:** Prepares processed audio frames for output by converting data to a suitable format.
  - **Channel Mapping:** Ensures that the left and right audio channels are correctly assigned, especially after any user-defined channel swap.
- **Role:**
  Acts as the interface between the digital processing world and the physical audio playback hardware. It ensures that the final, processed audio is delivered with fidelity and minimal latency.

Output to Playback Hardware

- **Components:**
  - **Hardware Interfaces:** Speakers, headphones, DAC, or sound cards.
- **Role:**
  This final stage converts the digital signals back into analog sound, enabling the user to hear the spatialized and filtered audio. It is the point where all the processing meets the human ear.

5. Key Concepts and Toolkits

3.1. Head-Related Transfer Function (HRTF)

- **Definition:**
  HRTF models how an acoustic signal is filtered by the human anatomy (head, torso, ears) before reaching the eardrum. This filtering is responsible for enabling spatial perception in listeners.
- **Usage in Code:**
  - The SOFA file (P0086_Raw_96kHz.sofa) is read by the sofaread function.
  - FIR filters (dsp.FIRFilter) are set up for the left and right channels using the HRTF data.
  - Slider controls update the HRTF coefficients dynamically, allowing for real-time alteration of the perceived sound source location.

3.2. DSP and Audio Playback

- **Digital Signal Processing:**
  - Uses FIR filtering for spatialization.
  - Employs Butterworth filters (via the butter function) for additional filtering scenarios.
- **Live FFT:**
  - The live FFT visualization computes the frequency domain representation of audio chunks to display spectral content.
  - A Hann window is applied for smoothing artifacts in the FFT output.
- **Audio Playback:**
  - audioDeviceWriter sends the processed audio to the playback device, ensuring real-time audio output synchronized with the GUI updates.

6. Challenges Faced During Development

Throughout the project, we encountered several significant challenges that ultimately pushed us to innovate and refine our approach:

- **Hunting for the Perfect SOFA Database:**
  One of our earliest challenges was locating a SOFA file that matched our exact use case. We scoured numerous databases, explored various forums, and sifted through countless resources before finally uncovering the ideal dataset. This journey not only taught us the importance of persistence but also deepened our understanding of spatial acoustics and HRTF data.

- **Mastering New MATLAB HRTF Functions:**
  Delving into the world of HRTF processing required us to master new MATLAB functions and techniques. We spent considerable time experimenting with different methods to accurately apply the HRTF filters and achieve lifelike spatial audio. This steep learning curve, while challenging, ultimately enriched our skill set and allowed us to leverage the full potential of MATLAB's capabilities.

- **Optimizing Real-Time Audio Processing:**
  Finding the optimal chunk size for processing was another complex hurdle. Early on, we experimented with various configurations and even considered using the Parallel Processing Toolkit. However, we soon discovered that parallelization introduced significant latency into the audio playback, which was unacceptable for real-time performance. This insight led us to develop an improved buffer management system, ensuring smooth and uninterrupted audio processing.

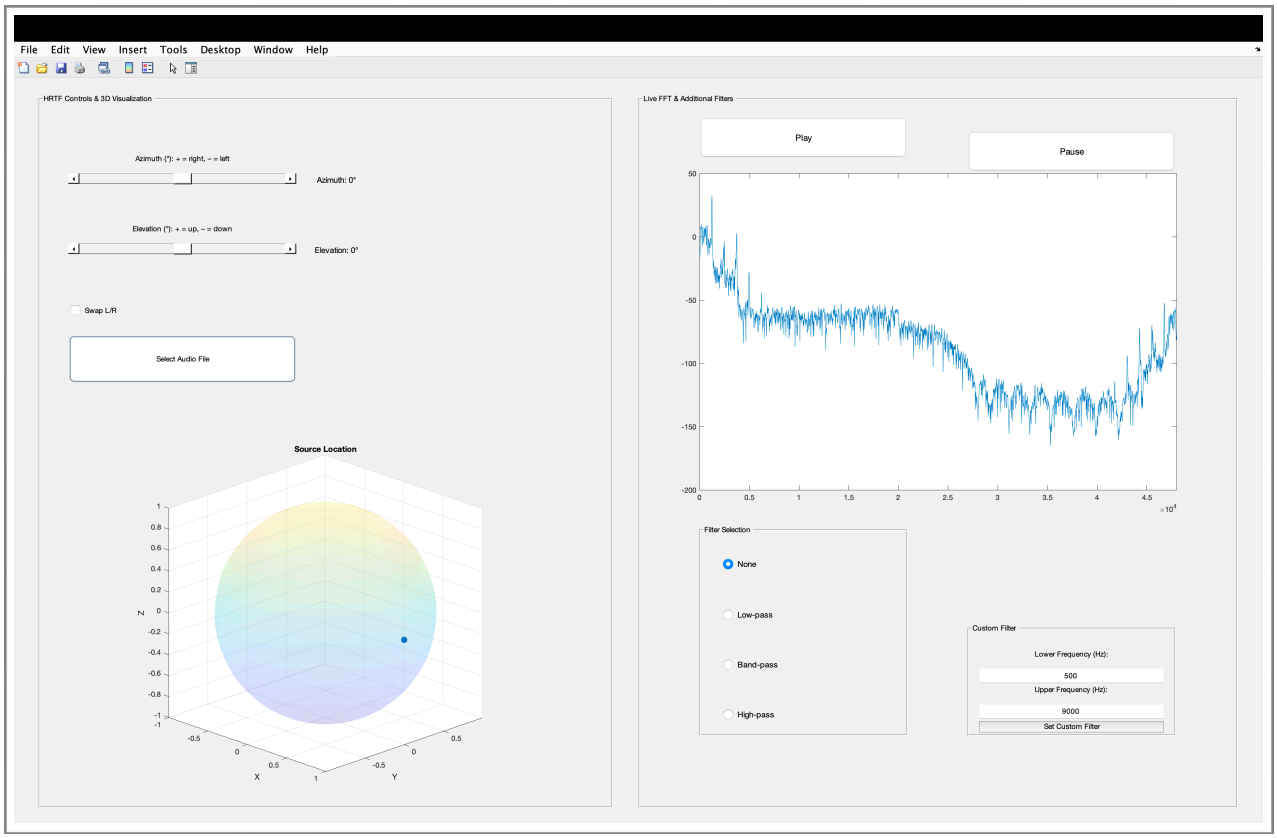
Each of these challenges played a pivotal role in shaping the final application, driving us to think creatively and work collaboratively to overcome obstacles along the way.

7. Images & Other Resources

4.1 GitHub Repository

https://github.com/BOSSX918spy/WaveSmith-DSP.git

4.2 Image Of Ui



8.

Conclusion

The provided script is a robust example of how MATLAB can be used to create an interactive audio processing application integrating advanced spatial audio techniques (via HRTF), live spectral analysis (FFT), and dynamic filtering capabilities.

Every component—from the GUI controls to the DSP routines and audio playback—has been carefully structured to ensure responsiveness, accurate state management, and efficient resource cleanup.

This detailed documentation should serve as a comprehensive reference for understanding and modifying the code as needed for further enhancements or integration with other audio processing projects.