《MATLAB 基础及图形用户界面设计》

课程设计

题目：音乐播放器实践

姓名：王子勋

学号：2024303578

班级：RB032401

学院：软件学院

日期：2025 年 3 月 2-9 日

目 录

1 摘 要

本次课程设计主要实现了音乐播放器的制作。播放器名为“睦头の音乐播放器”。该音乐播放器基于 Matlab App Designer GUI 开发平台进行设计，主要完成了音频文件的导入、播放、暂停、续播、停止、播放上一首、播放下一首、进度条显示等基础功能，并根据音频源文件实现了音量调整、速度调整、音乐风格切换、高级音效切换、人声消除、波形显示等功能，同时本人依托该音乐播放器的界面主题（乐队少女）进行了相关 UI 设计及彩蛋制作，在实现复杂功能的同时不失乐趣与美观性。

该音乐播放器为本人制作的第一个 App，对个人学习掌握 MATLAB 意义非凡。同时，对于使用者，该音乐播放器提供了丰富的可选择功能，既可作为休憩的方式，亦能通过界面 UI 及彩蛋感受惊喜与乐队少女文化。

以下为本人撰写的 App 英文概述：

instruction.txt

This app is the V1.0 version for my pratical work of MATLAB GUI course in the spring semester of 2025, guided by video of Zhiwei Guo.

The topic for my music player is the Japanese animation *BanG Dream! It's MyGO!!!!!* and *Ave Mujica*. From the inspiration of my roommate and my personal interest, I choose the character *Wakaba Mutsumi* of the animation as my main designment. In the music player, you can import music files from your personal PC and operate on them (i.e. play, pause, resume, stop, play previous piece of music and play next piece of music).

Addtionally, I implement following functions:

1. Music list and its display interface;
2. Music effect, including music styles (i.e. Jazz, Rock, Bass, etc.), playing speed, reverberation, high standard music effect (i.e. Concert, Live scene, chorus, etc.) and human voice cutting-down;
3. Time domain waveform in UI-axes;
4. Music playing process slider;
5. Music record rotivate and corresponding UI;
6. In-time volume adjustment;
7. Date display and the name of music playing display;
8. Menu control and additional notes;
9. Extra **surprise** options.

May you have a good time using the music player.

To enjoy music, to feel the joy and beauty in the girls' band, and to find surprise in the music player!

2 功能描述

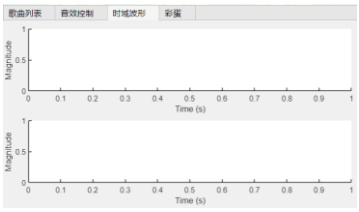
本次课程设计基于 Matlab App Designer GUI 开发平台，设计音乐播放器，实现音乐播放中的各项功能。最终实现设计的音乐播放器界面如图 2-1 及图 2-2 所示。

图 2-1 音乐播放器整体界面

（a）菜单

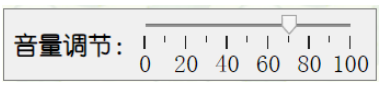
（b）播放列表

（c）音效控制



（d）时域波形

（e）唱片 UI



（f）音量调节

（g）彩蛋主界面

（h）版面切换　　（i）进度条及基本按钮

图 2-2 音乐播放器其他设置

2.1 功能 1：歌曲导入及歌曲列表显示

实现多个歌曲导入及歌曲列表显示的功能。点击【导入歌曲】按钮可以将多个歌曲进行一次导入，同时将导入的歌曲在歌曲列表中显示，如图 2-3 和图 2-4 所示。

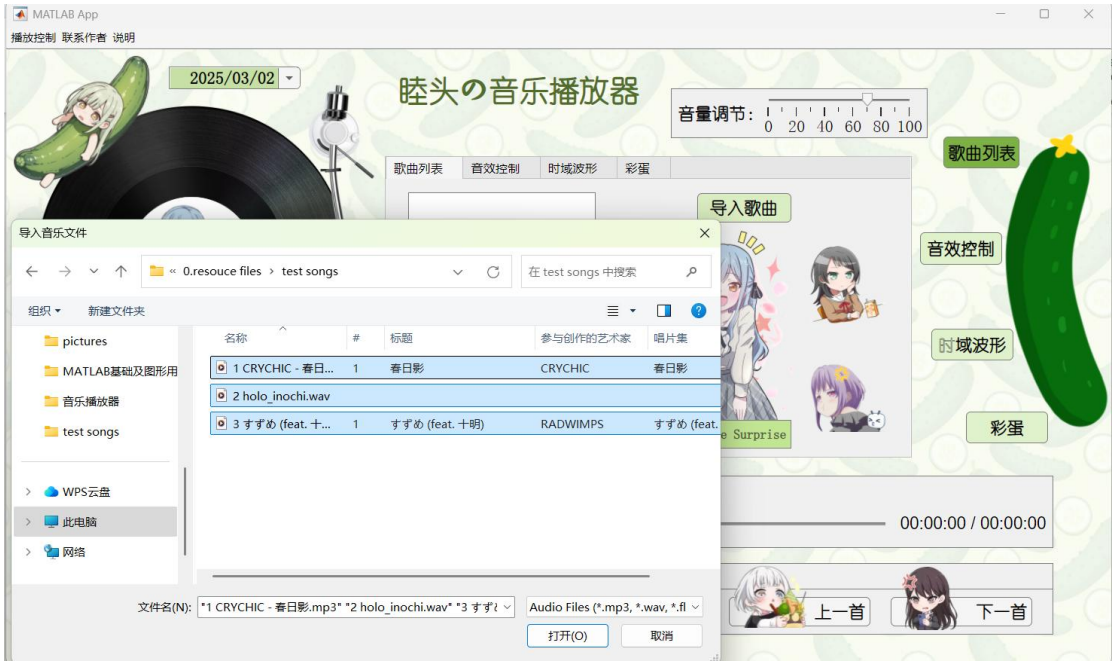


图 2-3 导入歌曲

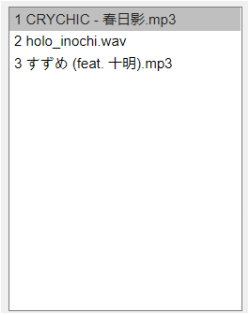


图 2-4 列表显示

2.2 功能 2：播放控制

实现音乐播放器中的多个播放控制功能，如图 2-5 所示。

图 2-5 播放控制选项

（1）**播　放：**点击后播放歌曲列表中当前选择的歌曲；

（2）**暂　停：**点击后暂停正在播放的歌曲；

（3）**续　播：**点击后从刚才暂停的地方继续播放歌曲；

（4）**停　止：**点击后停止播放歌曲；

（5）**上一首：**点击后播放当前歌曲的上一首歌曲；

（6）**下一首：**点击后播放当前歌曲的下一首歌曲。

在未导入歌曲状态下点击这 6 个按钮，将会报出警告提示框，如图 2-6 所示。

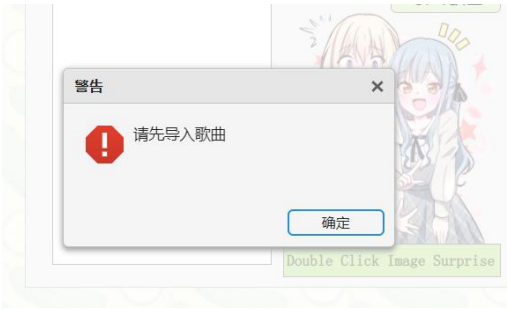


图 2-6 未导入歌曲警告

2.3 功能 3：音乐进度显示及控制

（1）**音乐进度显示：**音乐播放时进度条实时显示音乐播放的进度，进度条右侧显示当前播放的时刻及音乐总时长；

（2）**音乐进度 UI：**进度条滑块移动时 gif 图形随其移动；

（3）**音乐进度控制：**拖动进度条滑块到某个位置后，gif 图形移至同位，同时音乐调到该位置继续播放。

图 2-7 音乐进度显示及控制

2.4 功能 4：波形显示

播放时实时显示音乐文件时域波形，如图 2-8 所示。其中图 2-8 的下面的图显示的是当前播放的 2s 时间范围的音乐波形文件，图 2-8 的上面的图显示的是总时长范围内总波形图（黑色）以及当前播放的 2s 时间范围的波形（红色）。

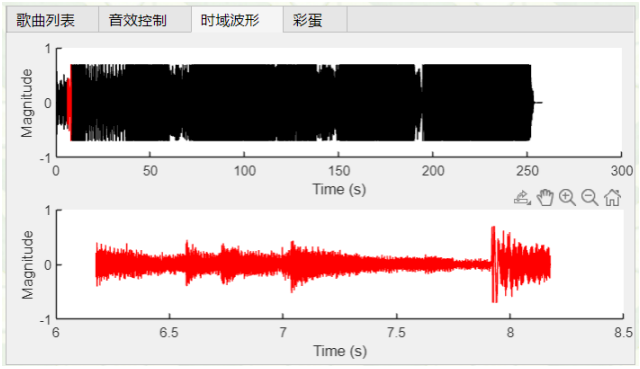


图 2-8 时域波形显示

2.5 功能 5：音效控制

音乐播放时可实现对音乐播放的音效控制，主要包括音乐风格、播放速度、混响、高级音效、人声消除等几个功能。

2.5.1 音乐风格

对音乐播放时的各频段增益进行调节（使用均衡器），实现 Normal、Jazz、Rock、Metal、Bass 等 5 种音乐风格的播放，如图 2-9 所示。

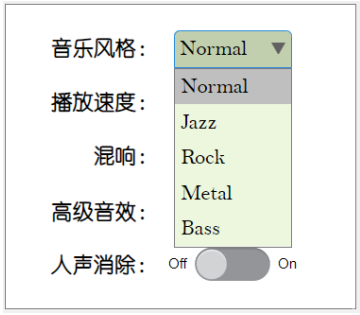


图 2-9 音乐风格

2.5.2 播放速度

实现快速、慢速、正常播放功能，包括 0.5 倍速、0.75 倍速、1 倍速、1.25 倍速、1.5 倍速、1.75 倍速、2 倍速播放功能，如图 2-10 所示。

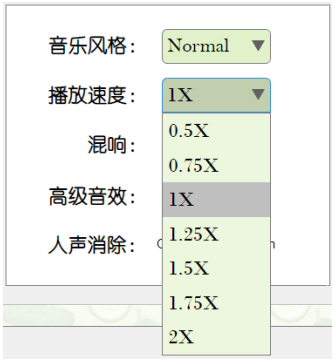


图 2-10 播放速度

2.5.3 混响

添加混响功能，使播放的声音具有混响的效果，混响时间参数可设置为 0.0s、0.04s、0.08s、0.12s、0.16s、0.2s，如图 2-11 所示。

图 2-11 混响设置

2.5.4 人声消除

消除人声，保留伴奏音乐，人声消除开关如图 2-12 所示。

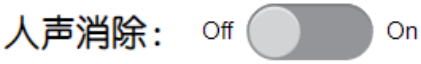


图 2-12 人声消除开关

2.5.5 高级音效

对音乐播放频段使用多种方式进行处理，实现音乐厅、电音、交响乐、Live 现场、HiFi 环绕、合唱、镶边、相位、立体声拓展、回声等 10 种高级音效的切换，如图 2-13 所示。

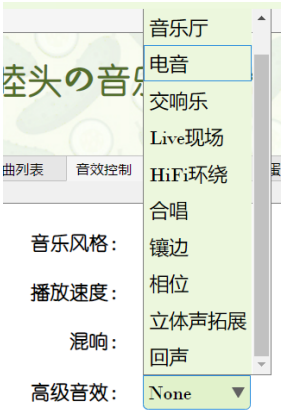


图 2-13 高级音效

2.6 功能 6：音量调节

播放时实时调节音量，如图 2-14 图 2-所示。默认音量 70%。

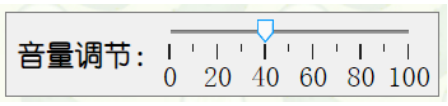


图 2-14 音量调节

2.7 功能 7：切换按钮及版面交互

初始状态 table 版面默认置于歌曲列表界面，如图 2-15 所示。通过点击 table 版面上方的切换框，或者点击最右侧黄瓜图样旁的按钮，可实现版面的切换，同时正在展示的界面对应的切换按钮显示深色，如图 2-16 所示。

图 2-15 音量调节

图 2-16 切换按钮功能展示

2.8 功能 8：本地日期显示

进入 App 后自动显示日期文字，如图 2-17 所示。

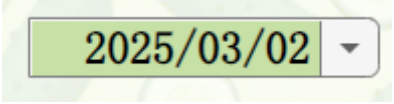


图 2-17 日期显示

2.9 功能 9：正在播放歌曲显示

进入 App 后正在播放提示文字显示为“None”，播放后切换为正在播放的歌曲名称，如图 2-18 所示。

图 2-18 正在播放歌曲文字提示在点击播放按钮前后变化

2.10 功能 10：唱片 UI 转换功能

进入 App 后唱片处于未播放状态，如图 2-19 所示。点击播放按钮，唱片置于播放状态，中心图片开始旋转，如图 2-20 所示。点击暂停或停止按钮，中心图片停止旋转，唱片置于未播放状态，同图 2-19。点击续播、上一首或下一首按钮，唱片继续置于播放状态。当再次置于播放状态的前一步为点击暂停按钮，唱片的中心图片将从再上一步停止的角度处继续旋转，如图 2-21；若为点击停止按钮，则从初始状态继续旋转。

图 2-19 未播放状态唱片 UI

图 2-20 播放状态唱片 UI

图 2-21 暂停后续播状态唱片 UI

2.11 功能 11：菜单显示

选中播放控制栏的 6 个子菜单中的一个，可同步实现对应名称播放按钮的控制，如图 2-22 所示。选中联系作者，点击学校邮件，将弹出本人邮件的信息提示对话框，如图 2-23 所示。选中说明，点击材料引用，将弹出 App 所引用的图片、音频、视频链接、视频的原出处的信息提示框，如图 2-24-1、2-24-2。

图 2-22 播放控制菜单

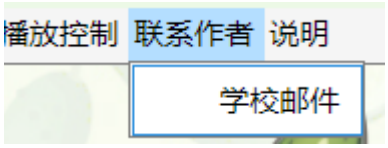


图 2-23-1 学校邮件选择菜单

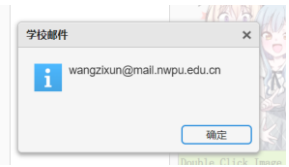


图 2-23-2 邮件提示框

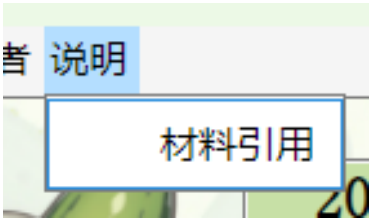


图 2-24-1 材料引用菜单

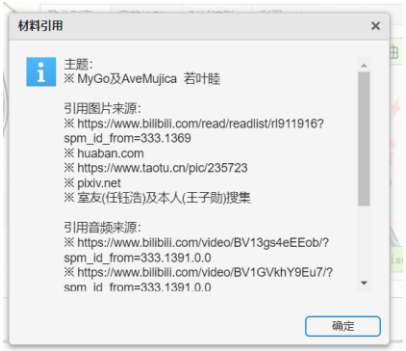


图 2-24-2 材料引用提示框

2.12 功能 12：基础彩蛋设计

在未播放状态下点击图片（几乎所有图片），将播放人物语音或歌曲。未播放状态指未导入歌曲状态，或导入歌曲后歌曲播放处于暂停或停止的状态。大部分人物语音会按内置顺序播放，点击一次播放一个语音，下一次在未播放状态点击时顺序播放下一个语音，随语音文件索引循环。在播放状态下点击人物语音无效。

此外，歌曲列表界面的中心大图具有双击彩蛋功能，即单击和双击播放的音频不同，双击播放的为一首歌曲（同人歌曲，原出处在 bilibili），当该歌曲处于播放状态下时，正在播放标签将显示该歌曲名称，其他键（除停止键）暂不可用，可待其播放完毕后恢复或点击停止键退出双击彩蛋，如图 2-25 所示。

彩蛋主界面左侧的图片外有三层圆环，处于播放状态下时，圆环的颜色会随机变化，圆环的大小随音乐波形大小按时域变化而起伏变化，如图 2-26 所示。

点击彩蛋主界面右下角的 1、2、3、5 标注的图片，会打开彩蛋视频链接，在电脑的默认浏览器内播放。点击 4 标注的图片，将弹出随机图片盲盒对话框，点击选取按钮可实现随机选择图片，点击退出可关闭该对话框，如图 2-27 所示。

图 2-25 双击彩蛋

图 2-26 圆环变化效果

图 2-27 图片盲盒对话框

2.13 功能 13：日期选择器隐藏彩蛋

日期选择器可以根据运行本地时间的日期，或者手动选择的日期，进入隐藏彩蛋。日期选择的标准是 MyGo 的 10 位主要女角色的生日。当本地运行时间为乐队少女的生日时，开始执行程序 table 组界面即显示隐藏彩蛋，并显示 label“今天是……的生日”。当手动修改日期选择器并满足选择日期为乐队少女生日时，进入隐藏彩蛋界面，并显示 label “…月…日是……的生日”。如图 2-28 所示。当所选日期非少女生日时，中央显示默认图片，如图 2-29 所示。

图 2-28 隐藏彩蛋界面

图 2-29 默认显示界面

视频播放前，中央显示当天生日的少女的图片，单击该图片，程序会用电脑默认图片处理工具打开该图片的高清大图。

主界面日期选择器旁有一个重置按钮，在隐藏彩蛋界面点击退出按钮后，只有点击重置按钮，程序才能重新进入隐藏菜单界面。

在隐藏彩蛋界面内，具有一个带彩蛋音频的人物图片、标签 label、视频播放及相应组件。点击“播放”会播放视频。基于计算机硬件处理能力的不同，视频的画面播放相对于视频的音频播放有延迟，故在视频上方放置了一个视频速度微调处理器，可实时更改帧率。点击“播放”后，播放按钮上的文字会更改为“暂停”，如图 2-30 所示。当选择暂停时，播放器会暂停视频画面及音频播放。此时播放按钮内文字会重置为“播放”。

图 2-30 视频播放

若点击停止按钮，视频播放器对象会清空。点击退出按钮即可退出该界面。

3 软硬件环境

3.1 运行环境

3.1.1 硬件环境

处理器型号：支持 SSE2 指令集的任何 Intel 或 AMD x86-64 处理器

内存容量：8GB 及以上

外存容量：SSD 硬盘，预留 20-30 GB 空间以容纳工具箱、数据和临时文件

3.1.2 软件环境

操作系统：64 位 Windows10 或 Windows11

应用软件：安装有 Matlab R2022b 或以上版本，且安装有 Matlab R2022b 对应的 Matlab Runtime，版本号 9.13。

3.2 开发环境

3.2.1 硬件环境

处理器型号：AMD Ryzen 7 7840H with Radeon 780M Graphics

内存容量：16GB

外存容量：1TB

3.2.2 软件环境

操作系统：64 位 Windows11

应用软件：Matlab R2022b。

4 主要功能实现方法简介

4.1 音乐播放器类及基础按钮

audioplayer 为 MATLAB 中专门用于处理音乐播放相关流程的类。对该类的创建方式如下：

```
MATLAB
player=audioplayer(Y,Fs);
player=audioplayer(Y,Fs,nBits);
player=audioplayer(Y,Fs,nBits,ID);
```

本次时间采用上述第一种创建方式，其为使用采样率 Fs 为信号 Y 创建 audioplayer 对象。该函数返回音频播放器对象 player。

音频信号 Y 指定为数值数据的向量或二维数组；采样频率 Fs 单位赫兹，指定为数值标量，大多数声卡支持的典型值有 8000Hz、11025Hz、22050Hz、44100Hz、48000Hz 和 96000Hz。

本次实践中所使用的 audioplayer 属性及函数如下：

SampleRate-采样频率：以赫兹为单位的采样频率，以数值标量形式返回。要设置 SampleRate，可在构造 audioplayer 对象时使用 Fs 输入参数。

TotalSamples-音频数据的总长度：此属性为只读，样本中音频数据的总长度，以整数形式返回。

CurrentSample-当前播放的样本：此属性为只读，是在音频输出设备播放的当前样本，以正整数形式返回。如果设备当前未播放音频，则 CurrentSample 是下一个要使用 play 或 resume 方法播放的样本。

TimerFcn-要重复执行的函数：在播放期间重复执行的函数，指定为包含函数名称的字符向量或字符串标量，或者指定为函数句柄。要指定重复时间间隔，需使用 TimerPeriod 属性。回调函数的前两个输入必须是 audioplayer 对象和 event 结构体。

TimerPeriod-计时器周期：指定为数值标量。计时器周期是 TimerFcn 回调的间隔时间（单位为秒）。

实现基础按钮功能（播放、暂停、续播、停止、上一首、下一首）时，需要使用以下对象函数：

play：从 audioplayer 对象播放音频。在播放按钮回调函数中的应用如下：

MATLAB　mutsumis_music_player.mlapp

```
% 播放按钮回调函数
function Button_playPushed(app, event)
    initPlayer(app); % 初始化音乐播放器对象
    if(~isempty(app.Player)) % 对象非空时
        play(app.Player, app.CurrentSample); % 播放
        app.isPlaying=true; % 播放判定置真

        % 显示音乐名称
        filePartIdx=find(app.ListBox_songs.Value == '.',1,'last');
        app.Label_name.Text=['正在播放：',app.ListBox_songs.Value(1:filePartIdx-1)];

        app.Image_record4.Visible="on";
        app.Image_record2.Visible="off";

        % 设置旋转标志
        app.isRotating = [true,false];
        rotateImg(app);
    else
        uialert(app.UIFigure, '请先导入歌曲', '警告');
        app.isPlaying=false; % 播放判定置假
    end
end
```

[1]

在 preMusic（上一首）与 nextMusic（下一首）相关回调函数中亦使用到 play 函数。以上函数实现了无导入歌曲时的点击提升警告框功能。

pause：暂停播放。在暂停按钮的回调函数中应用如下：

MATLAB　mutsumis_music_player.mlapp

```
% 暂停按钮回调函数
function Button_pausePushed(app, event)
```

```
    if(~isempty(app.Player)) % 对象非空时
        pause(app.Player); % 暂停
        app.isPlaying=false; % 播放判定置假
        filePartIdx=find(app.ListBox_songs.Value == '.',1,'last');
        app.Label_name.Text=['正在播放：',app.ListBox_songs.Value(1:filePartIdx-1)];

        app.Image_record4.Visible='off';
        app.Image_record2.Visible='on';

        % 设置旋转标志为 false
        app.isRotating = [false,false];
        % 记录当前旋转角度
        app.currentAngle = app.currentAngle;
    else
        uialert(app.UIFigure, '请先导入歌曲', '警告');
        app.isPlaying=false; % 播放判定置假
    end
end
```

resume：从暂停状态继续播放。调用方式为 resume(app.Player)，续播按钮回调函数与播放按钮回调函数类似。

stop：停止播放。调用方式为 stop(app.Player)，停止按钮回调函数与暂停按钮回调函数实现方法类似，并在其基础上增加了部分彩蛋功能。

4.2 播放速度控制

audioplayer 对象具有 SampleRate 属性，即采样频率。表示 1 秒时间内播放多少个点。使用 audioread 函数读取音频文件（[y, Fs] = audioread(filename)）后可获得该文件正常播放时的采样频率 Fs，正常播放时 audioplayer 的 SampleRate 值等于 Fs。

可以通过设定 audioplayer 的 SampleRate=x*Fs 改变播放器实际播放时的速度：

当 x=1 时，正常速度播放；

当 x<1 时，慢速播放；

当 x>1 时，快速播放。

函数实现如下：

MATLAB　mutsumis_music_player.mlapp

```
% 初始化音乐播放器
function initPlayer(app)
……
% 7.处理播放速度
    if(~isempty(app.Player))
        app.Player.SampleRate=round(app.Fs*app.PlaySpeed);
    end
end

% 速度播放按钮回调函数
function DropDown_speedValueChanged(app, event)
    value = app.DropDown_speed.Value;
    app.PlaySpeed=str2double(strip(value,'right','X')); % 获取速度值
    if(~isempty(app.Player)) % 对象非空时
        app.CurrentSample=app.Player.CurrentSample; % 将当前播放位置进行存储
        Button_playPushed(app,event); % 调用回调函数，立即播放
    end
end
```

[2]

4.3 音量控制

音量调节可以对音频文件的 SampleData 数据进行等比例缩放得到。先设置私有属性 Volume 对当前音量进行存储。在初始化播放器函数的设置如下：

MATLAB　mutsumis_music_player.mlapp

```
% 初始化音乐播放器
function initPlayer(app)
    ……
    % 5.处理音量
    app.SampleData=app.SampleData*app.Volume;
    ……
end
```

音量进度条的回调函数实现如下：

MATLAB　mutsumis_music_player.mlapp

```
% 音量进度条回调函数
function Slider_volumeAdjustValueChanged(app, event)
```

```
    value = app.Slider_volumeAdjust.Value;
    app.Volume=value/100; % 获取音量值
    if ~isempty(app.Player)
        app.CurrentSample=app.Player.CurrentSample; % 将当前播放位置进行存储
        Button_playPushed(app,event); % 调用回调函数，立即播放
    end
end
```

通过以上回调方法能实现动态改变音量。

4.4 混响设置

混响效果的产生来源于直达声与反射声的叠加，如图 4-1 所示为混响效果产生的示意图，听者所听到的声音不仅来自于直达声（DIRETO 路径），同时还来源于 R1、R2、R3、R4、R5 等反射路径反射的声音，叠加在一起就产生了混响的效果。

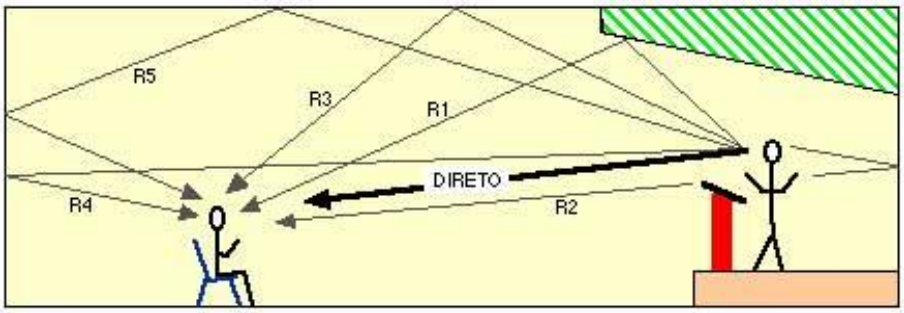


图 4-1 混响效果示意图

基于以上原理，可以获得一种在音乐播放器中产生混响的方法，通过将原始音频信号延时获得延时信号，再将延时信号与原始信号叠加产生最终播放时使用的音频信号，该音频信号含有混响的效果。由此编写 getReverbSampleData 函数获取添加混响效果后的音频数据，并存储在“getReverbSampleData.m”文件中，与.mlapp 主文件置于同一文件夹下，如下为该函数在主文件的调用方式及该函数代码。

MATLAB Mutsumis_Music_Player.mlapp

```
% 混响值下拉按钮回调函数
function DropDown_reverberationValueChanged(app, event)
    value = app.DropDown_reverberation.Value;
    app.ReverbValue=str2double(strip(value,'right','s')); % 获取混响值
    if(~isempty(app.Player)) % 对象非空时
        app.CurrentSample=app.Player.CurrentSample; % 将当前播放位置进行存储
```

```
            Button_playPushed(app,event); %  调用回调函数，立即播放
        end
end
```

MATLAB　getReverbSampleData.m

```
% 1.Function of this function
%         将音频数据加上混响效果，移位相加
% 2.Input parameters
%         sampleDataInitial:原始数据，输入为列向量
%         reverbValue:偏移时间
%         fs:采样频率
% 3.Output parameters
%         sampleData:加入混响效果后的数据
% 4.Examples
%         reverbValue=0.1;
%         sampleData=getReverbSampleData(sampleDataInitial, reverbValue);
% --------------Implementation goes here----------------------
function sampleData=getReverbSampleData(sampleDataInitial, reverbValue, fs)
        zeroNum=round(fs*reverbValue); %采样频率*延时时间=延时数据点数
        zerosData=zeros(zeroNum, size(sampleDataInitial, 2)); %延时处需要补零
        x0=sampleDataInitial; %原始音频数据
        x1=[x0; zerosData]; %原始音频数据+补零
        x2=[zerosData; x0]; %将原始数据向右偏移，左边补零，获得延时音频数据
        sampleData=(x1+x2)/2; %将原始数据与延时数据相加除以 2 获得混响数据
end
```

4.5 人声消除

4.5.1 简易人声消除

对于双声道的歌曲文件，一般情况下背景音乐在两个通道上的数据是有差别的，具有相位差；而人声在两个通道上的数据一般差别不大，是平均分配到两个声道上的。

基于这个原理，我们可以将左右声道的数据作差抵消来达到人声消除的效果。使用此种方法，有的音频文件人声消除效果较好，有的音频文件人声消除效果可能会较差，可能会影响背景音乐的清晰度。据此编写 cutDownHumanVoice 函数获取消除人声效果后的音频数据，存储在“cutDownHumanVoice.m”文件中，并

与.mlapp 主文件放置在同一个文件夹下。

函数如下所示。

MATLAB cutDownHumanVoice.m

```
% 1.Function of this function
%       将音频数据去除人声
% 2.Input parameters
%       sampleDataInitial:原始数据，输入为列向量，2 列（左右声道）
% 3.Output parameters
%       sampleData:去除人声效果后的数据
%       isSuccess:是否操作成功
% 4.Examples
%       [sampleData, isSuccess]=cutDownHumanVoice(sampleDataInitial);
% --------------Implementation goes here---------------------
function [sampleData, isSuccess]=cutDownHumanVoice(sampleDataInitial)
    if(size(sampleDataInitial, 2)==2) %双通道数据才进行操作
        deta=sampleDataInitial(:, 1)-sampleDataInitial(:, 2); %两个通道的差值
        sampleData(:, 1)=deta; %左通道赋值
        sampleData(:, 2)=deta; %右通道赋值
    else %单通道数据不进行操作
        sampleData=sampleDataInitial;
        isSuccess=false;
    end
end
```

4.5.2 基于 PCA 的人声消除

PCA 是一种降维技术，可以用于分离音频信号中的主要成分（如人声）和次要成分（如背景音乐）。

它通过线性变换将高维数据投影到低维空间，保留主要特征。其核心步骤如下：

（1）数据标准化

对数据进行标准化处理，使各特征均值为 0，方差为 1，公式为：

$$z = \frac{x - \mu}{\sigma}$$

图 4-2 数据标准化

其中，μ是均值，σ是标准差。

（2）计算协方差矩阵

协方差矩阵反映特征间的线性关系，计算公式为：

$$\Sigma = \frac{1}{n-1} X^T X$$

图 4-3 协方差矩阵

其中，X 是标准化后的数据矩阵。

（3）特征值分解

对协方差矩阵进行特征值分解，得到特征值和特征向量：

$$\Sigma = Q \Lambda Q^T$$

图 4-4 特征值分解

其中，Q是特征向量矩阵，Λ 是特征值对角矩阵。

（4）选择主成分

按特征值大小排序，选择前 k个最大特征值对应的特征向量作为主成分。

（5）数据投影

将数据投影到选定的主成分上，得到降维后的数据：

$$Y = XQ_k$$

图 4-5 数据投影

其中，Q_k是前 k个特征向量组成的矩阵。

通过 PCA，人声部分与背景音乐部分能得到较好的分离效果。而 MATLAB 中有关于 PCA 的内置函数。基于此，实现人声消除的步骤如下：

（1）将左右声道作为两个特征。

（2）对音频信号进行 PCA，提取主成分。

（3）去除第一个主成分（假设为人声），保留第二个主成分（假设为背景音乐）。

“cutDownHumanVoice.m”文件内的函数如下所示：

MATLAB cutDownHumanVoice.m

```
% 基于 PCA 进行人声消除
function [sampleData, isSuccess] = cutDownHumanVoice(sampleDataInitial)
    if size(sampleDataInitial, 2) ~= 2
        sampleData = sampleDataInitial;
        isSuccess = false;
        return;
```

```
        end

        % 1. 对左右声道进行 PCA
        [coeff, score, ~] = pca(sampleDataInitial);

        % 2. 去除第一个主成分（假设为人声）
        score(:, 1) = 0; % 将第一个主成分置零

        % 3. 重构信号
        sampleData = score * coeff';
        isSuccess = true;
    end
```

该函数在主文件中的调用方式如下：

MATLAB Mutsumis_Music_Player.mlapp

```
% 人声消除选择开关回调函数
function Switch_humanVoiceValueChanged(app, event)
    value = app.Switch_humanVoice.Value;
    if (strcmp(value,'Off')) % 不进行人声消除
        app.IsRemoveHumanVoice=false;
    else % 进行人声消除
        app.IsRemoveHumanVoice=true;
    end
    if(~isempty(app.Player))
        app.CurrentSample=app.Player.CurrentSample; % 将当前播放位置进行存储
        Button_playPushed(app, event); % 调用回调函数，立即播放
    end
end
```

4.6 音乐风格

如图 4-6 所示为音乐播放使用的均衡器，通过调节典型频率处的频响曲线，可获得不同的音乐风格。

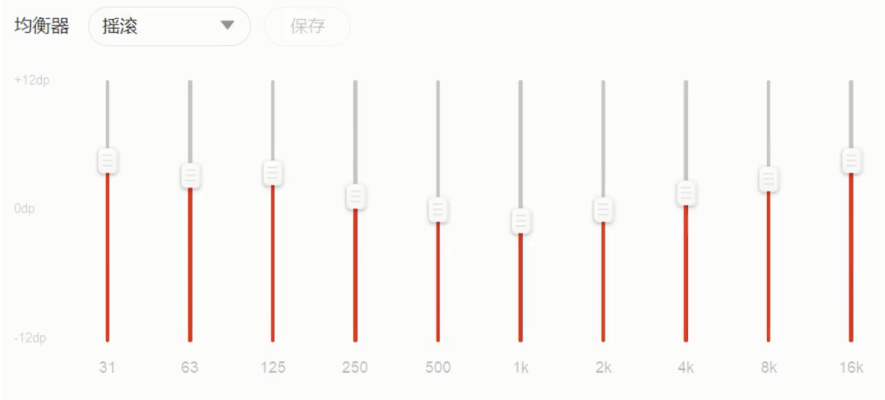


图 4-6 均衡器

使用代码实现频响曲线的调节主要采用滤波器滤波的方法，可以使用 Matlab 自带的 fir2 函数以及 filter 函数。不同的音乐风格对应不同的频响曲线，也就意味着不同的滤波器系数。设置均衡器的 setPopularEq 函数，将此函数保存在“setPopularEq.m”文件中，并与.mlapp 主文件放置在同一个文件夹下。

实现代码如下：

MATLAB setPopularEq.m

```
% 1.Function of this function
%       音频数据通过 Eq
% 2.Input parameters
%       sampleDataInitial:原始数据，输入为列向量，2 列（左右声道）
%       EqType:Eq 类型  'Normal', 'Jazz', 'Rock', 'Metal', 'Bass'
%       fs:采样频率
% 3.Output parameters
%       sampleData:  使用均衡器后的音频数据
% 4.Examples
%       sampleData=setPopularEq(sampleDataInitial, 'Jazz', fs);
% --------------Implementation goes here--------------------
function sampleData=setPopularEq(sampleDataInitial, EqType, fs)
fre=[0 31 63 125 250 500 1000 2000 4000 8000 16000 fs/2]; %12 个典型频率
f=fre/(fs/2); %归一化频率
if strcmp(EqType, 'Normal') %正常音效
    sampleData=sampleDataInitial;
else %附加其他音效
    switch EqType
        case 'Jazz' %爵士音效
            m=[0 1.995 1.995 1.259 1.585 0.631 0.794 1 1.259 1.585 2.512 0];
```

```
            case 'Rock' %摇滚音效
                m=[0 2.512 1.995 1.995 1.259 1 0.794 1 1.259 1.585 2.512 0];
            case 'Metal '%重金属音效
                m=[0 0.501 3.162 2.512 0.631 0.501 0.631 1.585 1.995 1.259 2.512 0];
            case 'Bass' %Bass 音效
                m=[0 3.981 3.162 6.310 1.585 1 1 1 1 1 1 0];
            otherwise
                m=[1 1 1 1 1 1 1 1 1 1 1 1];
        end
        b=fir2(100, f, m); %获得 FIR 滤波器系数
        sampleData=filter(b, 1, sampleDataInitial); %滤波
end
```

该函数在主文件中的调用方式如下：

MATLAB　Mutsumis_Music_Player.mlapp

```
% 音乐风格下拉选择按钮回调函数
function DropDown_musicStyleValueChanged(app, event)
    value = app.DropDown_musicStyle.Value;
    app.EqType=value; % 存储该 Eq 类型
    if(~isempty(app.Player)) % 对象非空时
        app.CurrentSample=app.Player.CurrentSample; % 将当前播放位置进行存储
        Button_playPushed(app,event); % 调用播放按钮的回调函数，立即播放
    end
end
```

4.7 播放进度的显示及控制

该音乐播放器可实现播放进度的显示及控制，如图 4-7 所示。

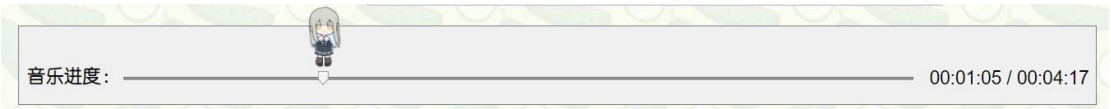


图 4-7 音乐进度显示及控制

4.7.1 进度条显示与控制

播放进度条所需的函数及实现方法如下。

（1）属性说明

aodioplayer 对象含有如下两个属性：

***TotalSamples —音频数据的总长度：**此属性为只读，样本中音频数据的总

长度，以整数形式返回。

***CurrentSample —当前播放的样本：**此属性为只读，在音频输出设备上播放的当前样本，以正整数形式返回。如果设备当前未播放音频，则 CurrentSample 是下一个要使用 play 或 resume 方法播放的样本。

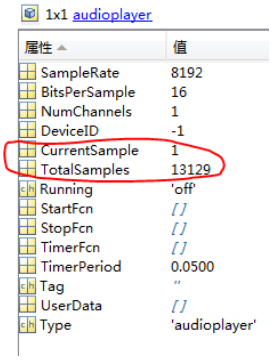


图 4-8 audioplayer 对象属性值

（2）音乐进度条的显示

音乐进度条的显示可以通过 CurrentSample/(TotalSamples+1)计算出当前播放的百分比，将此值设置为 App Designer 进度条的进度值（取值范围设为 0~1）即可。代码示意如下所示。

MATLAB Mutsumis_Music_Player.mlapp

```
% 计时器回调函数，实时更新进度条及波形显示
function Player_timeFcn(app)%
1.根据当前播放位置，设定进度条显示设置
    currentSample=app.Player.CurrentSample; % 当前播放到的数据点
    totalSamples=app.Player.TotalSamples; % 音乐文件总数据点数
    app.Slider_progress.Value=currentSample/(totalSamples+1); % 设定进度条当前值
    ……
end
```

（3）音乐进度条的控制

音乐进度条的控制指的是拖动进度条到某个位置后，音乐调到该位置继续播放。可以首先获取拖动后的进度条的进度值 value，然后使用 startPlaySample = (TotalSamples+1)*value 获取播放的起始点，使用 play 函数的如下形式即可：

play(playerObj, startPlaySample)：从 startPlaySample 所指示的样本至结尾播放音频。函数使用方式如下：

MATLAB Mutsumis_Music_Player.mlapp

```
% 进度条回调函数
function Slider_progressValueChanged(app, event)
```

```
    value = app.Slider_progress.Value;
    if~isempty(app.Player) % 当前播放器非空
        newSamplePos=floor((app.Player.TotalSamples+1)*value); % 获得新的起始播放点
数
        pause(app.Player); % 暂停当前播放
        play(app.Player,newSamplePos); % 从新的指定位置处播放
        ……
     else % 当前播放器为空
        app.Slider_progress.Value=0; % 进度条置 0
        ……
    end
end
```

4.7.2 Gif 图标随进度条移动效果

该效果具体描述如下：滑块随播放进度移动的同时，Gif 图标图标亦在滑块之上随其移动，手动拖拽进度条后，图标也到达新的位置。函数实现如下：

MATLAB Mutsumis_Music_Player.mlapp

```
% 计时器回调函数，实时更新进度条及波形显示
function Player_timeFcn(app)
    ……
% 4.gif 动图随时间移动
    currentGifPos=app.Image_gif.Position;
    app.Image_gif.Position=[121+769*app.Slider_progress.Value, ...
    currentGifPos(2),currentGifPos(3),currentGifPos(4)];
    ……
end
```

上述函数使用了 Image 组件的 Position 属性。该属性为 4 元的数组，依次为组件的 x、y 坐标及宽度、高度。函数中的 121 为原始 x 坐标，769 为进度条宽度由于进度条的 Value 属性属于[0,1]，故 121+769*app.Slider_progress.Value 即可表述计时器回调更新后的 x 坐标。[3]

4.8 时间格式变换

本实践后面在用文字显示音乐播放进度时，会需要将以秒计算的数据转换为以时分秒的格式显示，因此在此处编写 sec2HourMinSec 函数，保存在“sec2HourMinSec.m”文件中，并与.mlapp 主文件放置在同一个文件夹下。

MATLAB sec2HourMinSec.m

```
% 1.Function of this function
%       秒 变为 时：分：秒
% 2.Input parameters
%       tSecTotal:以秒表示的时间
% 3.Output parameters
%       tHourStr:时（两位字符串表示）
%       tMinStr:分（两位字符串表示）
%       tSecStr:秒（两位字符串表示）
% 4.Examples
%       Inputs:
%       tSecTotal=3670;
%       [tHourStr, tMinStr, tSecStr]=sec2HourMinSec(tSecTotal);
%       Results:
%       tHourStr='01'; tMinStr='01'; tSecStr='10';
% ----------Implementation goes here----------
function [tHourStr, tMinStr, tSecStr]=sec2HourMinSec(tSecTotal)
    tHour=floor(tSecTotal/3600); %取小时
    tSecTotalRemain=(tSecTotal-tHour*3600); %取完小时后，剩余的秒数
    tMin=floor(tSecTotalRemain/60); %取分钟
    tSecTotalRemain=(tSecTotalRemain-tMin*60); %取完分钟后，剩余的秒数
    tSec=floor(tSecTotalRemain); %取秒
    tHourStr=sprintf('%02d', tHour); %小时数转换成字符串输出
    tMinStr=sprintf('%02d', tMin); %分钟数转换成字符串输出
    tSecStr=sprintf('%02d', tSec); %秒数转换成字符串输出
end
```

4.9 绕定点旋转函数

音乐播放器的唱片 UI 需要使用到绕定点旋转函数，它需要使用仿射变换进行处理。

仿射变换是一种线性变换，可以表示为矩阵乘法。在二维空间中，仿射变换通常用于平移、旋转、缩放和剪切等操作。仿射变换的特点是保持直线的“直线性”和平行线的“平行性”。

仿射变换的一般形式为：

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

图 4-9 仿射变换一般形式

其中：(x,y)是原始坐标，(x',y')是变换后的坐标，3 阶矩阵是仿射变换矩阵。

该函数应用仿射变换的实现步骤如下：

（1）获取图片尺寸

获取输入图片的高度 h 和宽度w。

MATLAB rotateImageAroundPoint.m

```
[h, w, ~] = size(imageData);
```

（2）平移变换：将旋转中心点平移到原点

MATLAB rotateImageAroundPoint.m

```
tx = -center(1);
ty = -center(2);
translateToOrigin = [1, 0, 0; 0, 1, 0; tx, ty, 1];
```

旋转中心点 $(center(1),center(2))(center(1),center(2))$ 是用户指定的点。为了绕该点旋转，首先需要将该点平移到原点 (0,0)(0,0)。

平移变换矩阵为：

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ tx & ty & 1 \end{bmatrix}$$

图 4-10 平移变换矩阵

其中 $tx=-center(1)$，$ty=-center(2)$。

（3）旋转变换：绕原点旋转指定角度

MATLAB rotateImageAroundPoint.m

```
rotateMatrix = [cosd(angle), sind(angle), 0; -sind(angle), cosd(angle), 0; 0, 0, 1];
```

绕原点旋转角度θ（由 angle 指定）的变换矩阵为：

$$\begin{bmatrix} \cos(\theta) & \sin(\theta) & 0 \\ -\sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

图 4-11 中心旋转变换矩阵

（4）平移回原始位置

MATLAB rotateImageAroundPoint.m

```
translateBack = [1, 0, 0; 0, 1, 0; -tx, -ty, 1];
```

旋转完成后，需要将图片平移回原始位置。

平移变换矩阵为：

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -tx & -ty & 1 \end{bmatrix}$$

图 4-12 平移变换（至原始位置）矩阵

（5）组合变换

MATLAB rotateImageAroundPoint.m

```
tform = affine2d(translateToOrigin * rotateMatrix * translateBack);
```

将三个变换矩阵按顺序相乘，得到最终的仿射变换矩阵：

$$T_{\text{final}} = T_{\text{translateToOrigin}} \cdot T_{\text{rotate}} \cdot T_{\text{translateBack}}$$

图 4-13 矩阵组合变换

这个矩阵表示先平移、再旋转、最后平移回原始位置的组合变换。

（6）应用变换

MATLAB rotateImageAroundPoint.m

```
rotatedImage = imwarp(imageData, tform, 'OutputView', imref2d([h, w]));
```

使用 MATLAB 的 imwarp 函数对图片应用仿射变换。OutputView 参数指定输出图片的尺寸与输入图片相同（$h \times w$）。

函数编写并保存在“rotateImageAroundPoint.m”文件中，并与.mlapp 置于同一文件夹下。完整函数编写如下：

MATLAB rotateImageAroundPoint.m

```
% 1.Function of this function
%       绕指定点旋转图片
% 2.Input parameters
%       center: 图片旋转中心（图片左上角为原点）
%       imageData: 原始图片数据
%       angle: 旋转角度
% 3.Output parameters
%       rotatedImage: 旋转后的图片数据
% 4.Examples
%       rotatedImage = rotateImageAroundPoint(~, [100,100], imageData, 30)
% ----------Implementation goes here----------
function rotatedImage = rotateImageAroundPoint(~, center, imageData, angle)
    % 获取图片尺寸
```

```
    [h, w, ~] = size(imageData);

    % 将旋转中心点平移到原点
    tx = -center(1);
    ty = -center(2);

    % 创建平移变换矩阵
    translateToOrigin = [1, 0, 0; 0, 1, 0; tx, ty, 1];

    % 创建旋转变换矩阵
    rotateMatrix = [cosd(angle), sind(angle), 0; -sind(angle), cosd(angle), 0; 0, 0, 1];

    % 创建平移回原始位置的变换矩阵
    translateBack = [1, 0, 0; 0, 1, 0; -tx, -ty, 1];

    % 组合变换：平移 -> 旋转 -> 平移回
    tform = affine2d(translateToOrigin * rotateMatrix * translateBack);

    % 对图片进行旋转
    rotatedImage = imwarp(imageData, tform, 'OutputView', imref2d([h, w]));
end
```

4.10 高级音效设计

本人设计了 10 种音效，高级音效设计相关函数存储在“setHighQualityEq.m”文件中，与.mlapp 主文件置于同一文件夹下。

以下为高级音效主函数：

MATLAB　setHighQualityEq.m

```
% 1.Function of this function
%       音频数据通过高级音效
% 2.Input parameters
%       sampleDataInitial:原始数据，输入为列向量，2 列（左右声道）
%       EqType:Eq 类型  'None"音乐厅"电音"交响乐"Live 现场"HiFi 环绕"合唱"镶边"相位"立体声拓展"回声'
%       fs:采样频率
% 3.Output parameters
```

```
%       sampleData: 使用音效函数后的音频数据
% 4.Examples
%       sampleData=setHighQualityEq(sampleDataInitial, '音乐厅', fs);
% --------------Implementation goes here---------------------
function sampleData = setHighQualityEq(sampleDataInitial, EqType, fs)
    switch EqType
        case 'None'
            sampleData = sampleDataInitial;
        case '音乐厅'
            sampleData = setConcertHallEffect(sampleDataInitial, fs);
        case '电音'
            sampleData = setElectronicEffect(sampleDataInitial, fs);
        case '交响乐'
            sampleData = setOrchestraEffect(sampleDataInitial, fs);
        case 'Live 现场'
            sampleData = setLiveEffect(sampleDataInitial, fs);
        case 'HiFi 环绕'
            sampleData = setHiFiEffect(sampleDataInitial, fs);
        case '合唱'
            sampleData = setChorusEffect(sampleDataInitial, fs);
        case '镶边'
            sampleData = setFlangerEffect(sampleDataInitial, fs);
        case '相位'
            sampleData = setPhaserEffect(sampleDataInitial, fs);
        case '立体声拓展'
            sampleData = setStereoWideningEffect(sampleDataInitial);
        case '回声'
            sampleData = setEchoEffect(sampleDataInitial, fs, 0.3, 0.5);
    end
end
```

各音效的原理及函数实现如下：

4.10.1 音乐厅效果

音乐厅音效的特点是具有丰富的混响和空间感。

实现方法：

均衡器调整：增强中低频（200 Hz - 1 kHz）和高频（8 kHz 以上）。

混响效果：添加长混响，模拟音乐厅的空间感。

其中混响效果的实现除去应用前文所述 getReverbSampleData 函数，也可采用 MATLAB 自带的 reverberator 函数。其调用形式如下：

MATLAB setHighQualityEq.m

```
reverb = reverberator('PreDelay', 0, 'WetDryMix', 0.8, 'SampleRate', fs, 'DecayFactor', reverbTime);
```

参数说明：

PreDelay：预延迟时间（这里设置为 0，表示无延迟）。

WetDryMix：湿信号（混响信号）和干信号（原始信号）的混合比例（0.8 表示混响信号占主导）。

SampleRate：音频采样率。

DecayFactor：混响衰减因子，控制混响时间。

以下为音乐厅效果函数及辅助的混响效果函数：

MATLAB setHighQualityEq.m

```
% 1. 音乐厅音效
function sampleData = setConcertHallEffect(sampleDataInitial, fs)
    % 1. 均衡器调整
    fre = [0 31 63 125 250 500 1000 2000 4000 8000 16000 fs/2]; % 12 个典型频率
    f = fre / (fs/2); % 归一化频率
    m = [0 1.259 1.585 1.995 2.512 1.995 1.585 1.259 1.585 1.995 2.512 0]; % 音乐厅 EQ
    b = fir2(100, f, m); % 获得 FIR 滤波器系数
    sampleData = filter(b, 1, sampleDataInitial); % 滤波

    % 2. 添加混响效果
    reverbTime = 0.8; % 混响时间（0 到 1 之间）
    sampleData = addReverb(sampleData, fs, reverbTime);
end

% 混响效果
function output = addReverb(input, fs, reverbTime)
    % 限制 reverbTime 在 0 到 1 之间
    reverbTime = min(max(reverbTime, 0), 1);

    % 使用 MATLAB 的 reverberator 函数添加混响
    reverb = reverberator('PreDelay', 0, 'WetDryMix', 0.8, 'SampleRate', fs, 'DecayFactor',
```

```
reverbTime);
    output = reverb(input);
end
```

4.10.2 电音效果

电音音效的特点是强烈的低频和清晰的高频。

实现方法：

均衡器调整：增强低频（60 Hz 以下）和高频（8 kHz 以上）。

失真效果：添加轻微的失真效果，模拟电子音乐的特点。同时添加白噪声，模拟电流声，并添加调制效果。

失真效果的核心是通过非线性函数对音频信号进行处理，使信号的波形被“裁剪”或“压缩”，从而产生谐波失真。常见的非线性函数包括：

硬裁剪（Hard Clipping）：直接截断信号的幅值。

软裁剪（Soft Clipping）：使用平滑的非线性函数（如双曲正切函数）对信号进行处理。

此处使用双曲正切函数（tanh）来实现软裁剪。双曲正切函数 tanh (x)的数学表达式为：

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

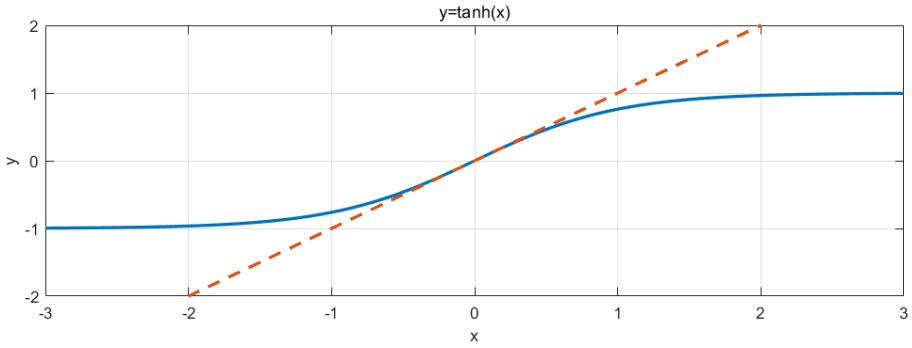


图 4-14 双曲正切函数及其图像

其特性如下：

当 x接近 0 时，tanh$(x)\approx x$（线性区域）。

当 x较大时，tanh (x)逐渐趋近于±1（饱和区域）。

这种特性使得双曲正切函数非常适合用于模拟软失真效果：小信号时，失真效果较弱，保留原始信号的特性；大信号时，失真效果明显，产生丰富的谐波。

失真效果辅助函数实现如下：

MATLAB setHighQualityEq.m

```
% 失真效果
```

```
function output = addDistortion(input, gain)
    % 更强的失真效果
    output = tanh(gain * input); % 使用双曲正切函数模拟失真
    output = output / max(abs(output(:))); % 归一化
end
```

白噪声效果可以由 randn 函数添加随机 SampleData 得到。

调制效果的核心是通过一个调制信号来改变输入信号的某些特性（如幅度、频率或相位）。此处使用的是幅度调制（AM），即通过调制信号改变输入信号的幅度。

幅度调制的一般形式为：

$$y(t) = x(t) \cdot [1 + m \cdot \cos(2\pi f_m t)]$$

图 4-15 幅度调制

其中：$x(t)$是输入信号（载波信号），m是调制深度（控制调制强度），f_m是调制信号的频率，$\cos(2\pi f_m t)$是调制信号。

调制效果辅助函数实现如下：

MATLAB setHighQualityEq.m

```
% 调制效果
function output = addModulation(input, fs)
    % 添加调制效果
    t = (0:length(input)-1)' / fs; % 时间向量
    modulator = sin(2 * pi * 5 * t); % 调制信号（5 Hz）
    output = input .* (1 + 0.5 * modulator); % 调制
end
```

以下为电音效果函数的主函数：

MATLAB setHighQualityEq.m

```
% 电音音效
function sampleData = setElectronicEffect(sampleDataInitial, fs)
    % 1. 均衡器调整
    fre = [0 31 63 125 250 500 1000 2000 4000 8000 16000 fs/2]; % 12 个典型频率
    f = fre / (fs/2); % 归一化频率
    m = [0 3.981 2.512 1.585 1.259 1 1 1.259 1.585 2.512 3.981 0]; % 电音 EQ
    b = fir2(100, f, m); % 获得 FIR 滤波器系数
    sampleData = filter(b, 1, sampleDataInitial); % 滤波
```

```
    % 2. 添加失真效果
    sampleData = addDistortion(sampleData, 0.5); % 更强的失真效果

    % 3. 添加白噪声（模拟电流声）
    noiseIntensity = 0.01; % 噪声强度
    noise = noiseIntensity * randn(size(sampleData)); % 生成白噪声
    sampleData = sampleData + noise;

    % 4. 添加调制效果
    sampleData = addModulation(sampleData, fs);
end
```

4.10.3 交响乐效果

交响乐音效的特点是平衡的频率响应和自然的空间感。

实现方法：

均衡器调整：保持频率响应平衡，略微增强中频（500 Hz - 2 kHz）。

混响效果：添加适度的混响，模拟音乐厅的环境。

混响效果实现同 4.8.1。以下为交响乐效果主函数：

MATLAB setHighQualityEq.m

```
% 3. 交响乐音效
function sampleData = setOrchestraEffect(sampleDataInitial, fs)
    % 1. 均衡器调整
    fre = [0 31 63 125 250 500 1000 2000 4000 8000 16000 fs/2]; % 12 个典型频率
    f = fre / (fs/2); % 归一化频率
    m = [0 1 1.259 1.585 1.995 2.512 1.995 1.585 1.259 1 1 0]; % 交响乐 EQ
    b = fir2(100, f, m); % 获得 FIR 滤波器系数
    sampleData = filter(b, 1, sampleDataInitial); % 滤波

    % 2. 添加混响效果
    reverbTime = 1.5; % 混响时间（秒）
    sampleData = addReverb(sampleData, fs, reverbTime);
end
```

4.10.4 Live 现场效果

Live 现场音效的特点是强烈的低频和清晰的中高频。

实现方法：

均衡器调整：增强低频（60 Hz 以下）和中高频（2 kHz - 8 kHz）。

延迟效果：添加轻微的延迟，模拟现场演出的空间感。

以下为 Live 现场效果的主函数及辅助的延迟函数：

MATLAB setHighQualityEq.m

```
% 4. Live 现场音效
function sampleData = setLiveEffect(sampleDataInitial, fs)
    % 1. 均衡器调整
    fre = [0 31 63 125 250 500 1000 2000 4000 8000 16000 fs/2]; % 12 个典型频率
    f = fre / (fs/2); % 归一化频率
    m = [0 3.162 2.512 1.995 1.585 1.259 1.585 1.995 2.512 3.162 2.512 0]; % Live EQ
    b = fir2(100, f, m); % 获得 FIR 滤波器系数
    sampleData = filter(b, 1, sampleDataInitial); % 滤波

    % 2. 添加延迟效果
    delayTime = 0.02; % 延迟时间（秒）
    sampleData = addDelay(sampleData, fs, delayTime);
end

% 延迟效果
function output = addDelay(input, fs, delayTime)
    % 添加延迟效果
    delaySamples = round(delayTime * fs); % 延迟的样本数
    output = [zeros(delaySamples, size(input, 2)); input(1:end-delaySamples, :)];
end
```

4.10.5 HiFi 环绕效果

HiFi 环绕音效的特点是宽广的声场和清晰的细节。

实现方法：

均衡器调整：保持频率响应平衡，略微增强高频（8 kHz 以上）。

立体声扩展：增强立体声效果，模拟环绕声场。

立体声拓展的核心是通过调整左右声道的增益（或相位）来增强声道的差异。常见的立体声拓展方法包括：

增益调整：通过增加一个声道的增益，同时减少另一个声道的增益，来增强声道的差异。

相位调整：通过改变一个声道的相位，使其与另一个声道产生相位差，从而增强空间感。

此处使用增益调整的方法。左声道信号被放大，右声道信号被缩小。这种不对称的增益调整会让人耳感觉到声音更偏向左侧，从而增强立体声的空间感。辅助的立体声拓展函数如下：

MATLAB　setHighQualityEq.m

```
% 立体声拓展
function output = addStereoWidening(input)
    % 增强左右声道的差异
    output = [input(:, 1) * 1.5, input(:, 2) * 0.5];
    output = output / max(abs(output(:))); % 归一化
end
```

HiFi 环绕效果主函数如下：

MATLAB　setHighQualityEq.m

```
% HiFi 环绕音效
function sampleData = setHiFiEffect(sampleDataInitial, fs)
    % 1. 均衡器调整
    fre = [0 31 63 125 250 500 1000 2000 4000 8000 16000 fs/2]; % 12 个典型频率
    f = fre / (fs/2); % 归一化频率
    m = [0 1 1 1 1 1 1 1 1.259 1.585 2.512 0]; % HiFi EQ
    b = fir2(100, f, m); % 获得 FIR 滤波器系数
    sampleData = filter(b, 1, sampleDataInitial); % 滤波

    % 2. 添加立体声扩展效果
    sampleData = addStereoWidening(sampleData);
end
```

4.10.6 合唱效果

合唱效果通过复制原始信号并对其进行轻微的延迟和调制，模拟多个声音同时播放的效果。

实现方法：复制原始信号并添加延迟和调制；将处理后的信号与原始信号混合。

合唱效果函数及其辅助函数如下：

MATLAB　setHighQualityEq.m

```
% 合唱音效
function sampleData = setChorusEffect(sampleDataInitial, fs)
    % 参数说明：
```

```
    % sampleDataInitial: 输入音频数据（双通道）
    % fs: 采样率

    % 1. 对左右声道分别处理
    leftChannel = sampleDataInitial(:, 1);
    rightChannel = sampleDataInitial(:, 2);

    % 2. 添加合唱效果
    leftChannelProcessed = addChorus(leftChannel, fs);
    rightChannelProcessed = addChorus(rightChannel, fs);

    % 3. 合并处理后的声道
    sampleData = [leftChannelProcessed, rightChannelProcessed];
end

function output = addChorus(input, fs)
    % 参数设置
    delay = 0.03; % 延迟时间（秒）
    depth = 0.005; % 调制深度
    rate = 0.5; % 调制频率（Hz）

    % 生成调制信号
    t = (0:length(input)-1)' / fs;
    modulator = depth * sin(2 * pi * rate * t);

    % 添加延迟和调制
    delayedSignal = [zeros(round(delay * fs), 1); input(1:end-round(delay * fs))];
    modulatedSignal = delayedSignal .* (1 + modulator);

    % 混合原始信号和处理后的信号
    output = input + modulatedSignal;
    output = output / max(abs(output)); % 归一化
end
```

4.10.7 镶边效果

镶边效果通过将延迟信号与原始信号混合，产生一种“飞行”或“旋转”的

声音效果。

实现方法：添加一个随时间变化的延迟信号；将延迟信号与原始信号混合。其实现方法与合唱效果类似，所不同的是镶边效果的延迟和调制范围较小，并且调制频率更低。

镶边效果及其辅助函数如下：

MATLAB　setHighQualityEq.m

```
% 镶边音效
function sampleData = setFlangerEffect(sampleDataInitial, fs)
    % 参数说明：
    % sampleDataInitial: 输入音频数据（双通道）
    % fs: 采样率

    % 1. 对左右声道分别处理
    leftChannel = sampleDataInitial(:, 1);
    rightChannel = sampleDataInitial(:, 2);

    % 2. 添加镶边效果
    leftChannelProcessed = addFlanger(leftChannel, fs);
    rightChannelProcessed = addFlanger(rightChannel, fs);

    % 3. 合并处理后的声道
    sampleData = [leftChannelProcessed, rightChannelProcessed];
end

function output = addFlanger(input, fs)
    % 参数设置
    delay = 0.01; % 基础延迟时间（秒）
    depth = 0.005; % 调制深度
    rate = 0.2; % 调制频率（Hz）

    % 生成调制信号
    t = (0:length(input)-1)' / fs;
    modulator = delay + depth * sin(2 * pi * rate * t);

    % 添加延迟和调制
```

```
    output = zeros(size(input));
    for i = 1:length(input)
        delaySamples = round(modulator(i) * fs);
        if i > delaySamples
            output(i) = input(i) + input(i - delaySamples);
        else
            output(i) = input(i);
        end
    end
    output = output / max(abs(output)); %  归一化
end
```

4.10.8 相位效果

相位效果通过将信号分成多个频段并对每个频段进行相位调制，产生一种“扫频”的声音效果。

实现方法：使用全通滤波器对信号进行相位调制；将处理后的信号与原始信号混合。相位效果的主要原理为动态调整全通滤波器的相位响应，产生“扫频”效果。即通过全通滤波器引入相位变化，然后将处理后的信号与原始信号混合，产生相消干涉和相长干涉。

全通滤波器是一种特殊的滤波器，其幅度响应在所有频率上都是平坦的（即不改变信号的幅度），但会改变信号的相位响应。全通滤波器的传递函数可以表示为：

$$H(z) = \frac{a + z^{-1}}{1 + az^{-1}}$$

图 4-16 全通滤波器传递函数

其中：z^{-1} 是单位延迟，a是滤波器的系数，控制相位响应的变化。

在时域中，全通滤波器的差分方程为：

$$y[n] = a \cdot x[n] + x[n-1] - a \cdot y[n-1]$$

图 4-17 全通滤波器差分方程

其中：$x[n]$是输入信号，$y[n]$是输出信号。

相位效果的实现方式如下：

（1）生成调制信号

调制信号是一个低频正弦波，用于动态调整全通滤波器的延迟时间。调制信号的频率决定了扫频的速度。

$$\text{modulator}[n] = \sin(2\pi f_m n / f_s)$$

图 4-18 调制信号生成

其中：f_m是调制频率（Hz），f_s是采样率（Hz），n是时间索引。

（2）动态调整延迟时间

延迟时间随调制信号的变化而变化：

$$\text{delay}[n] = \text{baseDelay} \cdot (1 + \text{modulator}[n])$$

图 4-19 延迟时间调整

其中：baseDelay 是基础延迟时间（秒），modulator[n] 是调制信号。

（3）全通滤波器的动态实现

在时域中，全通滤波器的差分方程可以改写为：

$$y[n] = a \cdot x[n] + x[n - \text{delay}[n]] - a \cdot y[n - \text{delay}[n]]$$

图 4-20 时域全通滤波器差分方程

其中：delay[n]是动态延迟时间（样本数）。

相位效果主函数及其辅助函数实现如下：

MATLAB　setHighQualityEq.m

```
% 相位音效
function sampleData = setPhaserEffect(sampleDataInitial, fs)
    % 参数说明：
    % sampleDataInitial: 输入音频数据（双通道）
    % fs: 采样率

    % 1. 对左右声道分别处理
    leftChannel = sampleDataInitial(:, 1);
    rightChannel = sampleDataInitial(:, 2);

    % 2. 添加相位效果
    leftChannelProcessed = addPhaser(leftChannel, fs);
    rightChannelProcessed = addPhaser(rightChannel, fs);

    % 3. 合并处理后的声道
    sampleData = [leftChannelProcessed, rightChannelProcessed];
end
```

```
function output = addPhaser(input, fs)
    % 参数设置
    rate = 0.5; % 调制频率（Hz）
    depth = 4; % 全通滤波器数量

    % 生成调制信号
    t = (0:length(input)-1)' / fs;
    modulator = sin(2 * pi * rate * t);

    % 全通滤波器
    output = input;
    for i = 1:depth
        output = allpassFilter(output, fs, modulator);
    end
    output = (input + output) / 2; % 混合原始信号和处理后的信号
    output = output / max(abs(output)); % 归一化
end

function output = allpassFilter(input, fs, modulator)
    % 全通滤波器实现
    gain = 0.7; % 增益

    output = zeros(size(input));
    for i = 1:length(input)
        % 动态调整延迟时间
        delay = round(0.001 * fs * (1 + modulator(i))); % 延迟时间随 modulator 变化
        if i > delay
            output(i) = gain * input(i) + input(i - delay) - gain * output(i - delay);
        else
            output(i) = input(i);
        end
    end
end
```

4.10.9 立体声拓展效果

立体声扩展通过增强左右声道的差异，使声音听起来更加宽广。

实现方法：对左右声道进行不同的处理；增强左右声道的差异。其效果为 HiFi 环绕音效的简化版，调用辅助函数 addStereoWidening 同 HiFi 环绕。

立体声拓展主函数如下：

MATLAB　setHighQualityEq.m

```
% 立体声拓展音效
function sampleData = setStereoWideningEffect(sampleDataInitial)
    % 参数说明：
    % sampleDataInitial: 输入音频数据（双通道）

    % 1. 添加立体声扩展效果
    sampleData = addStereoWidening(sampleDataInitial);
end
```

4.10.10 回声效果

回声效果通过添加延迟信号，模拟声音在空间中的反射。

实现方法：添加多个延迟信号；将延迟信号与原始信号混合。

回声效果主函数及其辅助函数如下：

MATLAB　setHighQualityEq.m

```
% 回声音效
function sampleData = setEchoEffect(sampleDataInitial, fs, delayTime, decay)
    % 参数说明：
    % sampleDataInitial: 输入音频数据（双通道）
    % fs: 采样率
    % delayTime: 延迟时间（秒）
    % decay: 衰减因子

    % 1. 对左右声道分别处理
    leftChannel = sampleDataInitial(:, 1);
    rightChannel = sampleDataInitial(:, 2);

    % 2. 添加回声效果
    leftChannelProcessed = addEcho(leftChannel, fs, delayTime, decay);
    rightChannelProcessed = addEcho(rightChannel, fs, delayTime, decay);
```

```
    % 3. 合并处理后的声道
    sampleData = [leftChannelProcessed, rightChannelProcessed];
end

function output = addEcho(input, fs, delayTime, decay)
    % 添加回声
    delaySamples = round(delayTime * fs);
    output = input;
    for i = delaySamples+1:length(input)
        output(i) = output(i) + decay * output(i - delaySamples);
    end
    output = output / max(abs(output)); % 归一化
end
```

4.11 图片盲盒自定义对话框

使用 uifigure 函数创建自定义对话框，再使用 uigreadlayout 函数调节网格布局。再依次添加 Image 组件、选取按钮与退出按钮。点击选取按钮需执行随机选图的自定义回调函数。图片在对话框中大小需要自适应比例变化。

将上述函数存储在“createImageDialog.m”文件中，并与.mlapp 主文件置于同一文件夹下。

MATLAB createImageDialog.m

```
% 1.Function of this function
%       根据输入图片文件创建对话框（主程序彩蛋 surprise4）
% 2.Input parameters
%       imageFiles: 图片路径名数组
% 3.Output parameters
%       none
% 4.Examples
%       createImageDialog(app.imageFiles);
% --------------Implementation goes here---------------------
function createImageDialog(imageFiles)
    % 创建对话框
    d = uifigure('Name', '图片盲盒', 'Position', [487 227 400 400], 'AutoResizeChildren', 'off');

    % 添加网格布局
```

```
    g = uigridlayout(d, [3 1]);
    g.RowHeight = {'1x', 'fit', 'fit'}; % 第一行自适应，第二行和第三行固定高度
    g.ColumnWidth = {'1x'}; % 单列，宽度自适应

    % 添加图片组件
    img = uiimage(g);
    img.Layout.Row = 1;
    img.Layout.Column = 1;
    img.Tag = 'DialogImage'; % 设置 Tag 以便后续访问

    % 添加“选取”按钮
    pickButton = uibutton(g, 'push', 'Text', '选取');
    pickButton.Layout.Row = 2;
    pickButton.Layout.Column = 1;
    pickButton.ButtonPushedFcn = @(~, ~) pickRandomImage(imageFiles, img, d);

    % 添加“退出”按钮
    exitButton = uibutton(g, 'push', 'Text', '退出');
    exitButton.Layout.Row = 3;
    exitButton.Layout.Column = 1;
    exitButton.ButtonPushedFcn = @(~, ~) delete(d);

    % 监听对话框大小变化
    d.SizeChangedFcn = @(~, ~) resizeImage(img, d);
end

function pickRandomImage(imageFiles, img, d)
    % 随机选择一张图片
    if isempty(imageFiles)
        disp('文件夹中没有图片！');
        return;
    end

    % 随机选择一个文件
    randomIndex = randi(numel(imageFiles));
```

```
    selectedFile = fullfile(imageFiles(randomIndex).folder, imageFiles(randomIndex).name);

    % 显示图片
    img.ImageSource = selectedFile;

    % 调整图片大小
    resizeImage(img, d);
end

function resizeImage(img, d)
    % 获取对话框的当前大小
    dialogWidth = d.Position(3);
    dialogHeight = d.Position(4);

    % 设置图片组件的大小（通过调整布局的行高和列宽）
    img.Parent.RowHeight{1} = dialogHeight - 100; % 第一行高度自适应
    img.Parent.ColumnWidth{1} = dialogWidth - 20; % 第一列宽度自适应
end
```

4.12 时域波形

通过(1:totalSamples)/app.Fs 可以得到时间点向量，通过对 app.SampleData 音频数据进行切片，可以截取单通道音频数据向量。利用 plot 函数，可以在坐标系上绘图。在绘制当前播放附近数据时，需注意接近结束时数据点的获取，以此确定起始索引。绘制总图时需要使用 hold 函数叠加展示当前播放数据，否则总图会被遮蔽。

波形显示相关函数如下：

MATLAB Mutsumis_Music_Player.mlapp

```
% 计时器回调函数，实时更新进度条及波形显示
function Player_timeFcn(app)
    ……
    % 3.波形显示
    % （1）获得总的数据
    tAll=(1:totalSamples)/app.Fs; % 从开始到结束所有时刻点
    tAll=tAll'; % 行向量变列向量
    xAll=app.SampleData(:,1); % 从开始到结束单通道中所有音频数据
```

```
    % （2）获得当前播放位置附近的显示区间的数据
    duraDisplay=2; % 当前播放位置附近显示的时长
    lengthDisplay=round(app.Fs*duraDisplay); % 当前播放位置附近显示的数据点数
    if currentSample+lengthDisplay-1<=totalSamples % 正常情况
        idxStart=currentSample; % 获得起始索引位置
        else % 显示区间段最后超过总数据点数，防止越界
        idxStart=totalSamples-lengthDisplay+1; % 获得起始索引位置
    end
    idxEnd=idxStart+lengthDisplay-1; % 获得显示的结束索引位置
    tDisplay=tAll(idxStart:idxEnd,1);
    xDisplay=xAll(idxStart:idxEnd,1);

    % （3）在时域波形 table 上面的坐标轴上绘图
    plot(app.UIAxes_all,tAll,xAll,'-k') % 绘制总图
    hold(app.UIAxes_all,'on') % 可叠加绘制开启
    plot(app.UIAxes_all,tDisplay,xDisplay,'-r') % 绘制当前播放位置附近的图
    hold(app.UIAxes_all,'off') % 可叠加绘制关闭

    % （4）在时域波形 table 下面的坐标轴上绘图
    plot(app.UIAxes_current,tDisplay,xDisplay,'-r') % 绘制当前播放附近的图
    ……
end
```

4.13 音频彩蛋

对每个需要添加彩蛋音频回调的图片，分别定义私有属性存储音频文件列表、当前播放的音频文件索引（每张图片对应彩蛋音频可能有多个）。在起始回调函数 startupFcn 中通过以下方式存储类似音频文件：

MATLAB　Mutsumis_Music_Player.mlapp

```
% 初始回调函数
function startupFcn(app)
    ……
    % 音频文件存储
    audioFolder = fullfile(app.appRoot, 'mygo audio', '若叶睦'); % 构建音频文件夹绝对路径
    files = dir(fullfile(audioFolder, '*.wav')); % 支持 .wav 文件
```

```
    files = [files; dir(fullfile(audioFolder, '*.mp3'))]; % 支持 .mp3 文件
    app.audioGif = fullfile(audioFolder, {files.name}); % 写入音频文件路径数组
    ……
end
```

其中 app.appRoot 为主文件根路径私有属性。[4]app.audioGif 类似形式的属性为对应彩蛋存储音频文件列表的私有属性。

在类似图片回调函数的实现如下：

MATLAB Mutsumis_Music_Player.mlapp

```
% Image_gif 回调函数
function Image_gifClicked(app, event)
    % 彩蛋
    if ~app.isPlaying
        offButtonAndImage(app);
        app.Image_gif.Enable="on";

        currentAudioFile = app.audioGif{app.curIdGif};

        % 读取音频文件
        [y, fs] = audioread(currentAudioFile);

        % 播放音频
        sound(y, fs);

        onButtonAndImage(app);

        % 更新索引以指向下一个音频文件
        app.curIdGif = app.curIdGif + 1;

        % 如果索引超出范围，则重置为 1
        if app.curIdGif > numel(app.audioGif)
            app.curIdGif = 1;
        end
    end
end
```

其中 isPlaying 是 app 对象的私有属性（前文 4.1 亦有叙述），在点击按钮 play、

resume、preMusic、nextMusic 时（app.Player 非空）置 true，在点击 pause 及 stop 时置 false，其初始化为 false。offButtonAndImage(app)与 onButtonAndImage(app)为对象 app 的私有函数，其实现方式是对所有具有彩蛋的图片的 Enable 属性赋'off'或'on'，实现执行一个音频彩蛋回调函数时其他按键不可用。此处采用简单的 sound 函数播放音频。

4.14 双击回调效果

通过本地点击时间戳间隔来判断是否为双击，此处设置为 0.5 秒内连击即通过双击判定。判定函数如下：

MATLAB Mutsumis_Music_Player.mlapp

```
% 双击判定
function doubleClick=ImageClicked(app)
    % 获取当前时间
    currentTime = datetime('now');
    currentTime = posixtime(currentTime); % 转换为时间戳（秒）

    % 计算时间间隔
    timeInterval = currentTime - app.lastClickTime;

    % 判断是否为双击
    if timeInterval < 0.5 % 双击时间间隔阈值（0.5 秒）
        doubleClick=true;
    else
        doubleClick=false;
    end

    % 更新上一次点击时间
    app.lastClickTime = currentTime;
end
```

其中 app.lastClickTime 属性初始置 0，函数内部随点击时刻变化实现递归调用。datetime 函数可以对当前时间（'now'）及当前日期（'today'）等进行读取，在日期选择器回调函数中也要使用。

通过双击判定后，对应回调函数方能执行双击的彩蛋效果。

4.15 圆环时域起伏效果

[5]彩蛋主界面的圆环通过读取时间序列上的音频 SampleData 进行一定比例的半径变化，因而产生随音频播放而起伏的效果。圆环的颜色采用随机冷色调，边界颜色为该冷色调的补色。

在计时器回调函数中的实现方式如下：

MATLAB Mutsumis_Music_Player.mlapp

```
% 计时器回调函数，实时更新进度条及波形显示
function Player_timeFcn(app)
    ……
    % 5.彩蛋 circle
    minSample=min(xAll);
    maxSample=max(xAll);

    cla(app.UIAxes_circle);
    hold(app.UIAxes_circle, 'on');

    % 定义新的圆环参数
    theta = linspace(0, 2*pi, 100); % 角度
    r1 = 90 + floor((xAll(currentSample,1)-minSample)*50/(maxSample-minSample));
    r2 = 0.96*r1;
    r3 = 80 + floor((xAll(currentSample,1)-minSample)*40/(maxSample-minSample));
    r4 = 0.97*r3;
    r5 = 70 + floor((xAll(currentSample,1)-minSample)*30/(maxSample-minSample));
    r6 = 0.98*r5;

    % 计算外圆和内圆的坐标
    x_outer = r1 * cos(theta);
    y_outer = r1 * sin(theta);
    x_inner = r2 * cos(theta);
    y_inner = r2 * sin(theta);

    x2_outer = r3 * cos(theta);
    y2_outer = r3 * sin(theta);
    x2_inner = r4 * cos(theta);
    y2_inner = r4 * sin(theta);
```

```
    x3_outer = r5 * cos(theta);
    y3_outer = r5 * sin(theta);
    x3_inner = r6 * cos(theta);
    y3_inner = r6 * sin(theta);

    % 关闭坐标轴自动缩放
    app.UIAxes_circle.XLimMode = 'manual';
    app.UIAxes_circle.YLimMode = 'manual';

    % 设置坐标轴范围
    xlim(app.UIAxes_circle, [-140, 140]); % X 轴范围
    ylim(app.UIAxes_circle, [-140, 140]); % Y 轴范围

    % 绘制圆环
    color=app.colors(randi([1,10]),1:3);
    color1=app.colors(randi([1,10]),1:3);
    color2=app.colors(randi([1,10]),1:3);
    fill(app.UIAxes_circle, [x_outer, fliplr(x_inner)], [y_outer, fliplr(y_inner)], color,
'EdgeColor', [1,1,1]-color); % 填充圆环
    fill(app.UIAxes_circle, [x2_outer, fliplr(x2_inner)], [y2_outer, fliplr(y2_inner)], color1,
'EdgeColor', [1,1,1]-color1); % 填充圆环
    fill(app.UIAxes_circle, [x3_outer, fliplr(x3_inner)], [y3_outer, fliplr(y3_inner)], color2,
'EdgeColor', [1,1,1]-color2); % 填充圆环
    axis(app.UIAxes_circle, 'equal'); % 设置坐标轴比例相等

     % 隐藏坐标轴和边框
     app.UIAxes_circle.Color=[0.94,0.94,0.94];
     app.UIAxes_circle.XAxis.Visible = 'off'; % 隐藏 X 轴
     app.UIAxes_circle.YAxis.Visible = 'off'; % 隐藏 Y 轴
     app.UIAxes_circle.Box = 'off'; % 隐藏边框
     ……
end
```

以上函数中定义了 3 个同心圆环。xAll 为波形显示时读取的 SampleData 数据向量。颜色为[R,G,B]三元有序数组。注意一定要在代码界面关闭坐标轴自动缩

放，设计界面关闭效果不佳。在给定区间内，根据当前 SampleData 减去最小 SampleData 的差占 SampleData 极差的比例计算出当前半径大小。

上述函数中的 app.colors 为颜色二维数组私有属性，储存了一些冷色调颜色。

MATLAB Mutsumis_Music_Player.mlapp

```
properties (Access = private)
    ……
    colors = [
        0.5294, 0.8078, 0.9216; % 天蓝色
        0, 1, 1;                % 青色
        0, 0.5019, 0.5019;      % 蓝绿色
        0, 0, 0.5019;           % 深蓝色
        0.5961, 1, 0.5961;      % 薄荷绿
        0.8, 0.8, 1;            % 蓝紫色
        0.6784, 0.8471, 0.9020; % 冰蓝色
        0, 0.3922, 0;           % 深绿色
        0.2510, 0.8784, 0.8157; % 青绿色
        0.4157, 0.3529, 0.8039  % 灰蓝色
    ] % 冷色调卡
    ……
end
```

4.16 日期选择器隐藏彩蛋

日期选择器的回调函数会返回 value 值，通过 month()、day()函数可以读取选择日期的月、日。根据 value 与乐队少女生日的比较，可以给出界面显示的 label 及 image。

MATLAB 使用 VideoReader 类对视频进行读取（只能读取画面，音频仍需 audioplayer 类读取）。并且视频需要依托 UIAxes（作为父容器的子容器）进行显示。调用方法为：app.videoReader=VideoReader(videoFilePath)。视频播放按钮的回调函数如下：

MATLAB Mutsumis_Music_Player.mlapp

```
% 播放按钮回调
function Button_videoPlayPushed(app, event)
    app.Image_birth.Visible='off';
    if isempty(app.videoReader)
        videoFile=fullfile(app.appRoot,'video','video1.mp4'); % 视频绝对路径
```

```
        app.videoIndex=1;
        app.videoReader=VideoReader(videoFile); % 创建视频读取对象
        [app.vAudioData, app.vSampleRate]=audioread(videoFile);
        app.audioVideoPlayer=audioplayer(app.vAudioData,app.vSampleRate);
        app.videoPlaying=true;
        app.videoPaused=false;
        app.Button_videoPlay.Text="暂停";
        play(app.audioVideoPlayer);

        while app.videoPlaying && hasFrame(app.videoReader)
            if ~app.videoPaused
                frame=readFrame(app.videoReader);
                imshow(frame, 'Parent', app.UIAxes_video);
                lora=app.Spinner_lora.Value;
                pause(1/(app.videoReader.FrameRate*lora));
            else
            pause(0.1); % 短暂暂停以避免占用过多 CPU
        end
end

% 播放结束
if ~app.videoPaused
        app.videoPlaying=false;
        app.Button_videoPlay.Text='播放';
end
elseif app.videoPlaying && ~app.videoPaused && app.videoIndex==1
% 暂停播放
        app.videoPaused=true;
        app.Button_videoPlay.Text='续播';
        pause(app.audioVideoPlayer); % 暂停音频
elseif app.videoPlaying && app.videoPaused && app.videoIndex==1
% 续播视频
        app.videoPaused=false;
        app.Button_videoPlay.Text='暂停';
        resume(app.audioVideoPlayer); % 恢复音频
```

```
        end
    end
```

由上可知，视频播放由 readFrame()函数读取帧，imshow()函数展示画面，pause()函数暂停极小时间形成播放效果。暂停时间由 app.videoReader 的属性 FrameRate 决定，同时暂停时间要乘以 1/lora 进行微调，处理器不同而微调参数不同（见 2.13）。

使用系统自带的图像处理工具打开图片时，根据系统类型不同，使用函数不同，实现效果如下：

MATLAB Mutsumis_Music_Player.mlapp

```
% 单击打开图片回调函数
function Image_birthClicked(app, event)
    ……
    systemType = computer;
    switch systemType
        case {'PCWIN', 'PCWIN64'}    % windows64 系统
            winopen(birthPng);
        case 'MACI64'    % mac64 系统
            system(['open "', birthPng, ""]);
        case 'GLNXA64'    % linux64 系统
            system(['xdg-open "', birthPng, ""]);
        otherwise
            error('不支持的操作系统: %s', systemType);
    end
end
```

5 实现效果展示

5.1 App 打包

5.1.1 桌面 App 打包

在.mlapp 设计界面上方点击设计工具，选择 App 详细信息，对 App 的信息进行编辑，如图 5-1 所示。

图 5-1 App 详细信息编辑

编辑完成后，在设计工具栏选择共享，点击独立桌面 App，进行编辑。

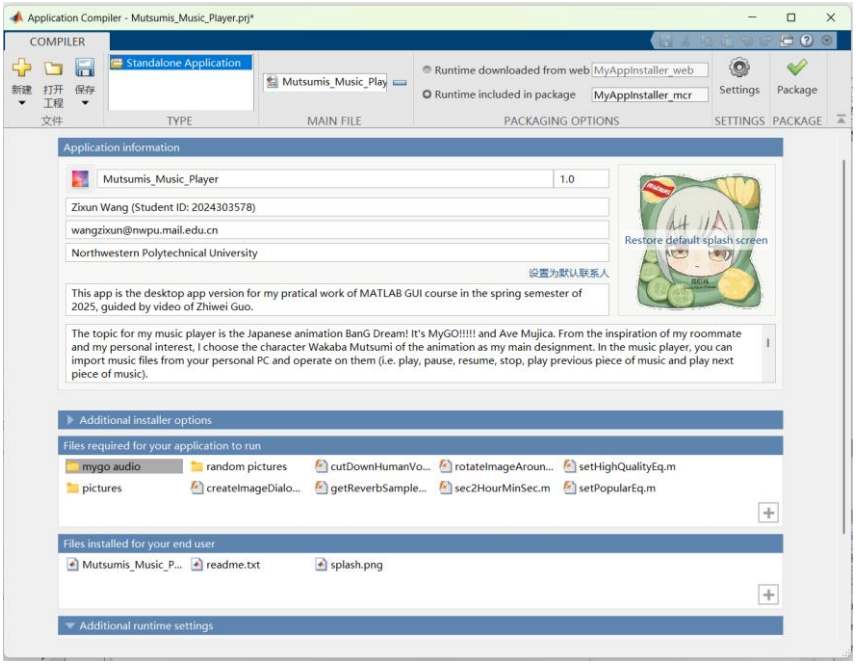


图 5-2 独立桌面 App 编辑

需要注意将所有相关程序及辅助文件（音频、图片文件）导入。选择 Runtime included in package（MATLAB runtime 通过预设项 MATLAB compiler 进行下载）。在命令行窗口输入 mcrinstaller，确认 runtime 是否已经下载。

图 5-3 命令行 runtime 指令

可自主添加图标。最后选择 Package，即可进行打包。

图 5-4 导出文件夹

导出文件夹见图 5-4，由上至下分别为用户使用 App 下载程序、开发者使用 App 下载程序、开发者测试 App 下载程序。

执行 for_redistribution 文件夹里的安装程序，在安装界面修改安装路径[6]等选项进行安装。

5.1.2 MATLAB App 打包

在 5.1.1 编辑的基础信息的基础上，在设计工具栏选择共享-MATLAB App。

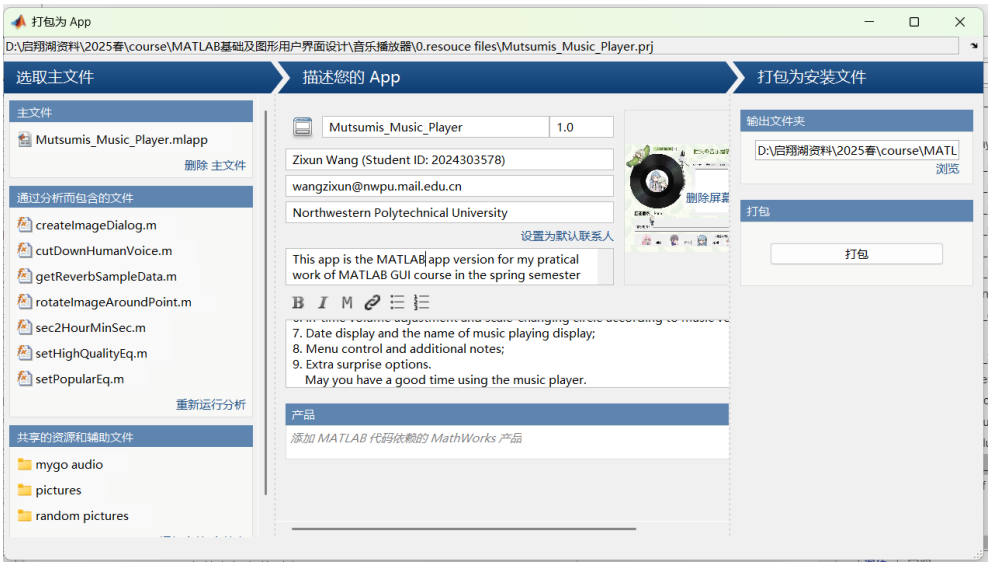


图 5-5 MATLAB App 编辑

注意编辑时添加共享的资源和辅助文件。编辑完点击“打包”即可。

打包成功后，在 MATLAB 主页找到导出文件夹，并将其设置为当前文件夹，右键音乐播放器安装程序，选择安装，即可将 MATLAB App 导入我的 App。

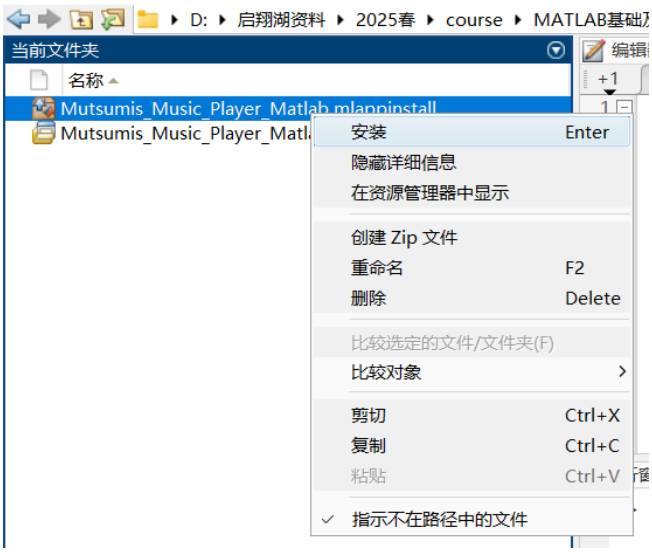


图 5-6 安装 MATLAB App

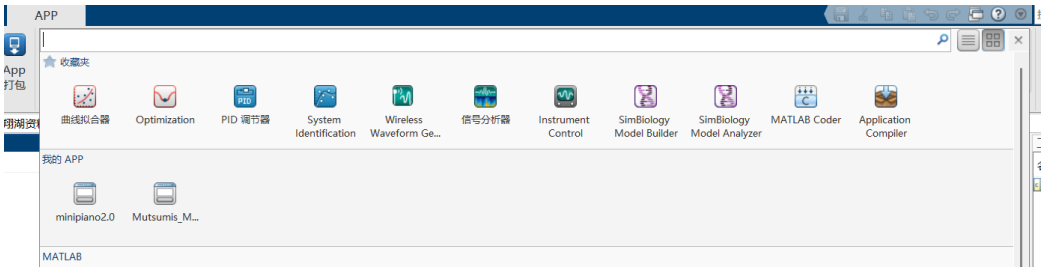


图 5-7 我的 App 显示界面

5.1.3 Web App 打包

在 5.1.1 编辑的基础信息的基础上，在设计工具栏选择共享-Web App。

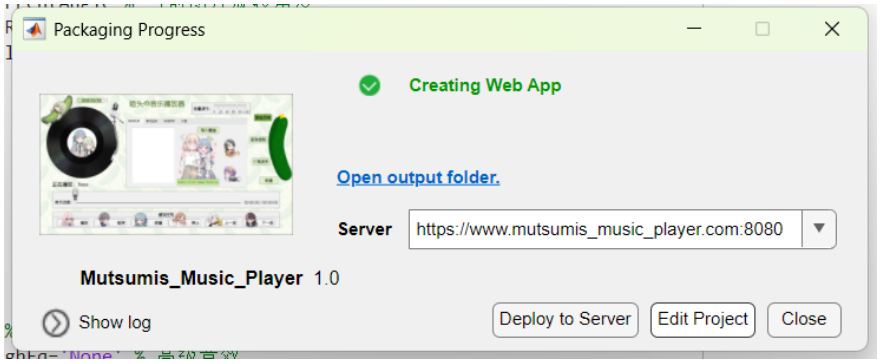


图 5-8 Packaging Progress 界面

弹出界面如图 5-8 所示。点击 Edit Project。进入信息编辑界面，如图 5-9 所示。在 Files required for your app to run 中添加必要的函数文件及需读取的音频、

图片文件夹。

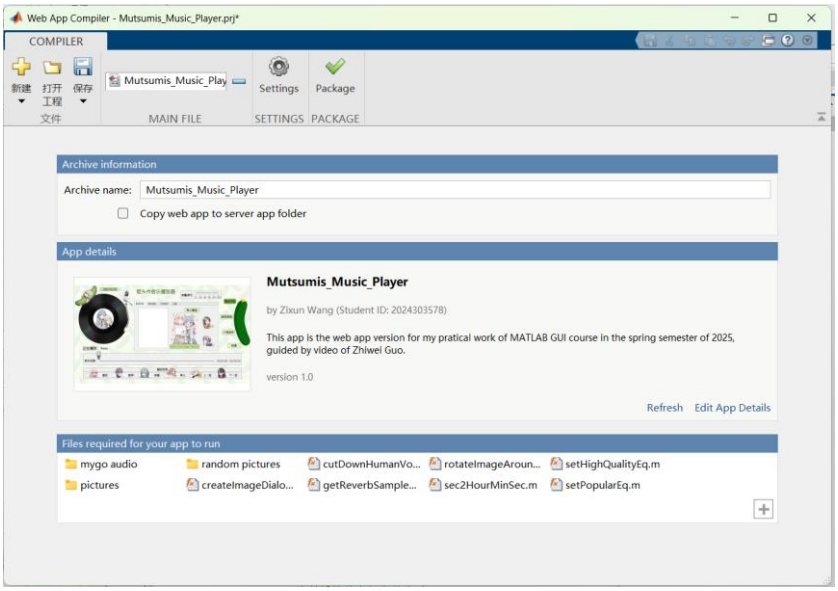


图 5-9 Web App 编辑

选择 Package，即可打包。

在以下类似路径下找到 MATLAB Web App Server 安装程序并安装：

文件资源管理器
D:\MATLABR2022b\MATLAB\toolbox\compiler\deploy\win64\MATLABWebAppServerSetup

安装完成后，打开 MATLAB Web App Server，设置相关参数，选择 Open App Folder，将打包生成的.ctf 文件复制入该文件夹。选择 Open Home Page，即可在默认浏览器中进入 Web App 主页，如图 5-11 所示。其中的红框内为计算机设备名称。

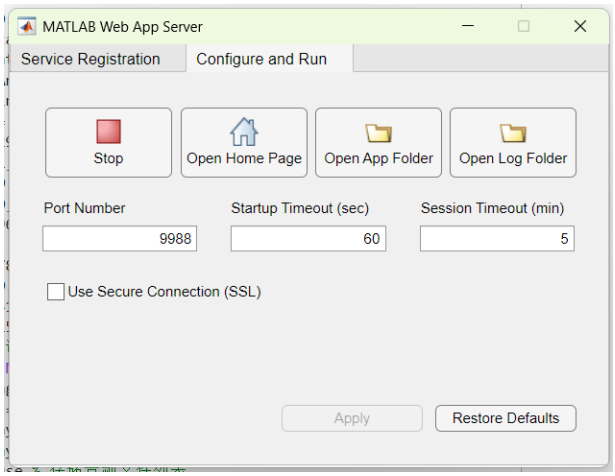


图 5-10 MATLAB Web App Server 程序

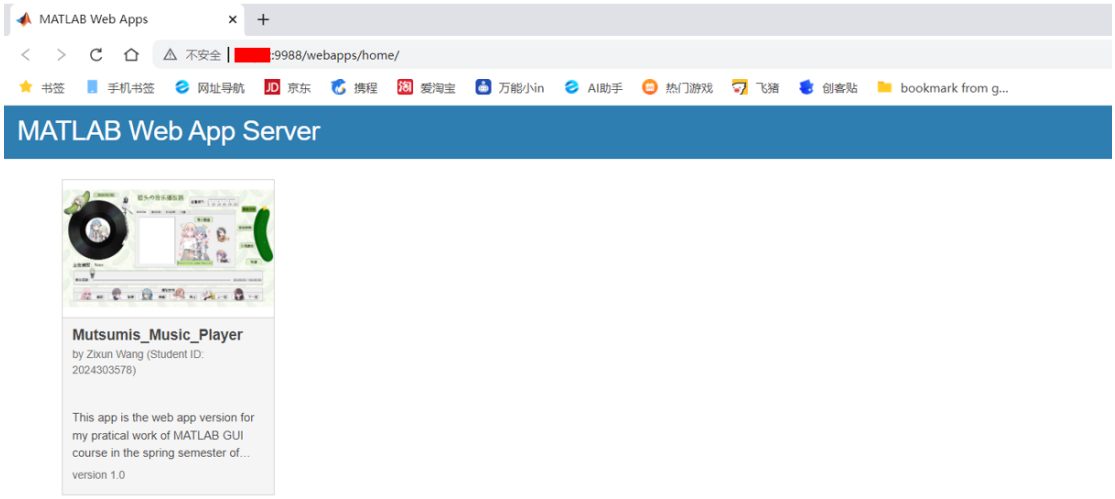


图 5-11 MATLAB Web App Server Home Page

5.2 效果展示

独立桌面 App 与 MATLAB App 的展示界面相同，如图 5-12 所示。

图 5-12 App 运行主界面

以下为部分运行效果图，更多功能在第 2 章里有详细描述及相应界面展示。


MATLAB App
播放控制 联系作者 说明
2025/03/06
睦头の音乐播放器
音量调节：
0 20 40 60 80 100
歌曲列表
音效控制
时域波形
彩蛋
1 CRYCHIC-春日影.mp3
2 holo-inochi.wav
3 すずめ (feat. 十明).mp3
4 DiverDiva-Eternal Light.flac
5 MIMINARI _ 橘木灯-眠れない (难
6 Lezel-未完成ランデヴー (未完成的
7 EGOIST-エウテルペ (欧式耳偶).fla
8 三无Marblue-神のまにまに (神的随
导入歌曲
Double Click Image Surprise
歌曲列表
音效控制
时域波形
彩蛋
正在播放：8 三无Marblue-神のまにまに（神的随…
音乐进度：
00:00:20 / 00:04:14
播放控制
播放
暂停
续播
停止
上一首
下一首


MATLAB App
播放控制 联系作者 说明
2025/03/06
睦头の音乐播放器
音量调节：
0 20 40 60 80 100
歌曲列表
音效控制
时域波形
彩蛋
音乐风格：
Normal
播放速度：
1.25X
混响：
0.04s
高级音效：
音乐厅
人声消除：
Off
On
歌曲列表
音效控制
时域波形
彩蛋
正在播放：8 三无Marblue-神のまにまに（神的随…
音乐进度：
00:00:23 / 00:04:14
播放控制
播放
暂停
续播
停止
上一首
下一首


MATLAB App
播放控制 联系作者 说明
2025/03/06
睦头の音乐播放器
音量调节：
0 20 40 60 80 100
歌曲列表
音效控制
时域波形
彩蛋
音乐风格：
Normal
播放速度：
1X
混响：
0.0s
高级音效：
None
人声消除：
Off
On
歌曲列表
音效控制
时域波形
彩蛋
正在播放：8 三无Marblue-神のまにまに（神的随…
音乐进度：
00:00:48 / 00:04:14
播放控制
播放
暂停
续播
停止
上一首
下一首

图 5-13 效果展示组图

Web app 打开 Home Page 进入 App，展示界面如下：

图 5-14 Web App 展示界面[7]

6 实践总结及感想

以下为实践过程中的问题及对相应上文的注解：

[1]：app.isplaying 也是音乐播放器对象的函数，用于确定播放是否正在进行。我原先打算使用该函数用于部分彩蛋功能，但在调试及执行过程中发现 play 函数执行完毕后会返回初始播放点，app.isplaying 函数始终置真，无法进入条件语句内部。暂时未找到很好的解决方法。目前采用自定义的 app 私有属性 isPlaying，在回调函数执行过程时根据执行操作置真假。

[2]：在学习郭志巍老师的音乐播放器录播视频时，主函数中对于 speed 方法的使用方式如下：

MATLAB　mutsumis_music_player.mlapp

```
% 初始化音乐播放器
function initPlayer(app)
……
% 7.处理播放速度
    app.Player.SampleRate=round(app.Fs*app.PlaySpeed);
end
……
```

即未在 app.Player.SampleRate 前对 app.Player 是否非空进行判定，后期本人在警告框效果的处理上遇上判定播放器是否为空的情况（点击播放相关的 6 个按钮，播放器为空则弹出警告框），运行时发现未导入歌曲时未弹出警告框，经调试，app.Player 在初始化播放器时，其 SampleRate 属性便被赋值，致使 app.Player 初始非空。通过添加 if 判断语句，问题得以解决。

[3]：在编写 Gif 图标随进度条滑块移动效果的函数时，我曾将函数写为如下形式：

MATLAB　Mutsumis_Music_Player.mlapp

```
% 计时器回调函数，实时更新进度条及波形显示
function Player_timeFcn(app)
    ……
% 4.gif 动图随时间移动
    currentGifPos=app.Image_gif.Position;
    app.Image_gif.Position=[currentGifPos(1)+769*app.Slider_progress.Value, ...
    currentGifPos(2),currentGifPos(3),currentGifPos(4)];
    ……
```

```
end
```

在运行时发现图标随时间移动并未保持在滑块上方，更进一步观察运行界面，图标随时间位移呈平方增长。本人使用 disp 函数在命令行打印 app.Image_gif 的 Position(1)进行调试，通过数学计算推出 currentGifPos(1)随时间变化，即在计时器函数调用过程中 app.Image_gif 的 Position 属性会动态更新，因此我将 currentGifPos(1)修改为初始 x 坐标。

[4]：最开始程序打包前，我并没有定义 app 的私有属性 appRoot。原始音频文件存储程序如下：

MATLAB Mutsumis_Music_Player.mlapp

```
% 初始回调函数
function startupFcn(app)
    ……
    % 音频文件存储
    audioFolder ='mygo audio\若叶睦'; % 构建音频文件夹相对路径
    files = dir(fullfile(audioFolder, '*.wav')); % 支持 .wav 文件
    files = [files; dir(fullfile(audioFolder, '*.mp3'))]; % 支持 .mp3 文件
    app.audioGif = fullfile(audioFolder, {files.name}); % 写入音频文件路径数组
    ……
end
```

以上使用的是音频文件夹相对路径，在 MATLAB 程序中能正常执行。但在打包为 app 后（打包过程中所需文件夹已导入），发现报错，无法通过原始相对路径读取音频文件夹，经互联网检索知，MATLAB 中程序经打包后相对路径会发生变化，于是添加 appRoot 属性读取主文件所在绝对路径，实现方式如下：

MATLAB Mutsumis_Music_Player.mlapp

```
properties (Access = private)
    ……
    appRoot = fileparts(mfilename('fullpath')) % 获取当前主文件的目录
    …….
end
```

[5]：设计圆环时域起伏效果的灵感来自于 Wallpaper Engine 的 scene 场景类型壁纸的音频监听功能，在壁纸上具有更精细的效果，此处实现的是劣化版本。本音乐播放器圆环起伏不连贯的原因在于个人 PC 的 MATLAB 的多线程执行能力有限，计算资源及效率不足以供应程序采集到较精细的时间序列样本。同样出现不

连贯问题的还有唱片中心旋转效果。经检验，同时执行的效果有 3 个及以上（比如同时播放音乐、旋转唱片、打开彩蛋说明提示框）时，程序会出现较大的停顿。操作过于频繁也会导致某些回调函数不能完全执行，致使一些私有属性的赋值未及时更替（比如私有属性 isPlaying、isRotating）。

受限于本地 MATLAB 的不止时间序列样本采集，还有对 png 格式图像的处理。在将 png 图片转换为可供旋转操作的格式时，MATLAB 并不支持 α 图层（即透明图层）的数据读取，仅能读取 3 个列向量（R、G、B）。

[6]：不知何原因，我的安装界面并没有将应用快捷键发送到桌面上的选项。

[7]：在 Web App 中，不知何原因，图形界面发生了位移。此外，在当前版本浏览器（联想自带浏览器及 Google Chrome）不支持 audioread 等函数，致使功能无法实现。因而实际上 Web App 并没有完成。如下为日志报错提示图片。

图 6-1 Web App 报错

当然，在独立桌面 App 及 MATLAB App 中也有一些问题。比如在 MATLAB App 中，关闭程序后，某些变量没有成功删除，从而让命令行窗口报错。

本次实践中，我通过学习郭志巍老师的录播视频，集中学习了 MATLAB App Designer 的相关设计知识，并在老师的示范基础上创新，如 UI 界面图形设计、彩蛋设计等，提高了对 MATLAB 的掌握程度，制作成人生中的第一个 App，并体会到程序自主开发设计的乐趣。在编写过程中，我遇到不少问题，都尽力通过调试、AI 工具、查阅书籍等手段进行修正与弥补。在某些设计通过 AI 工具进行优化的同时，我也学到了相当多数学与计算机科学知识，比如 PCA、全通滤波器

等。此外，我通过大量的注释完整诠释了程序编写的思路，逐渐养成良好编程习惯。希望我能在接下来的学习中更进一步，以面向对象的思想设计更多有效益的程序。

7 课程建议

本人希望郭志巍老师能对录播视频进行更新，尽量使用较新版本的MATLAB，并对同学们在实践过程中遇到的问题进行搜集与提示（如我在运行Web App 时遇到的 audioplayer 使用问题）。

此外，我希望老师上课时能在将基础知识的同时与实际应用相结合，比如在讲到矩阵时便可以展示一下它在线性代数的线性变换、旋转变换中的应用。