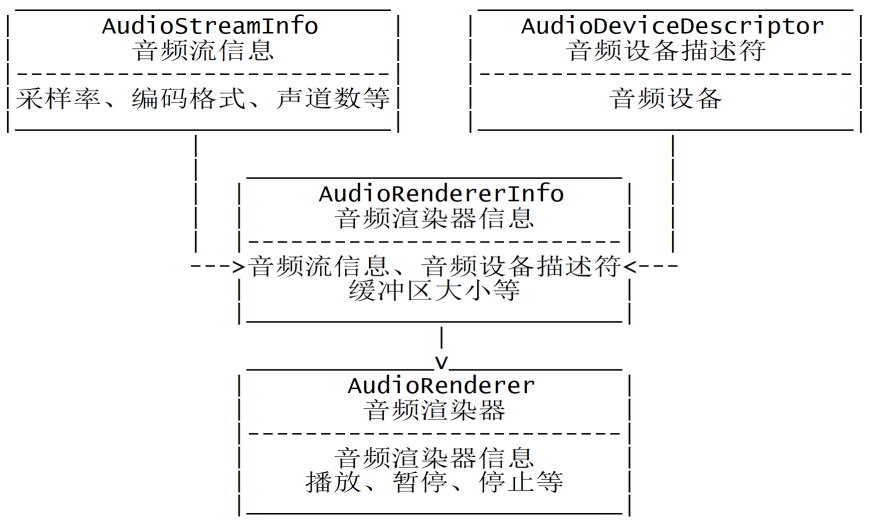
多媒体

1 音频渲染

1.1 创建音频渲染器

使用鸿蒙系统提供的音频渲染器播放音频文件(资源)需要用到以下四个对象：



1.1.1 创建音频流信息

```
// 采样率
int sampleRate = 44100;
// 音频流标志
AudioStreamFlag audioStreamFlag =
    AudioStreamFlag.AUDIO_STREAM_FLAG_DIRECT_OUTPUT;
// 编码格式
EncodingFormat encodingFormat = EncodingFormat.ENCODING_PCM_16BIT;
// 声道数
ChannelMask channelMask = ChannelMask.CHANNEL_OUT_STEREO;
// 使用类型
StreamUsage streamUsage = StreamUsage.STREAM_USAGE_MEDIA;

// 音频流信息
AudioStreamInfo audioStreamInfo =
    new AudioStreamInfo.Builder()
        .sampleRate(sampleRate)
        .audioStreamFlag(audioStreamFlag)
        .encodingFormat(encodingFormat)
        .channelMask(channelMask)
        .streamUsage(streamUsage)
        .build();
```

1.1.2 获取音频渲染器缓冲区大小

```
// 获取音频渲染器缓冲区大小
bufferSizeInBytes = AudioRenderer.getMinBufferSize(
    sampleRate, encodingFormat, channelMask);
```

1.1.3 获取音频设备描述符

```
private AudioDeviceDescriptor getSpeaker() {
    AudioDeviceDescriptor speaker = null;

    // 获取音频输出设备列表
    AudioDeviceDescriptor[] devices =
        AudioManager.getDevices(DeviceFlag.OUTPUT_DEVICES_FLAG);

    // 查找扬声器设备
    for (AudioDeviceDescriptor device : devices)
        if (device.getType() == DeviceType.SPEAKER) {
            speaker = device;
            break;
        }

    return speaker;
}
```

1.1.4 创建音频渲染器信息

```
// 音频渲染器信息
AudioRendererInfo audioRendererInfo =
    new AudioRendererInfo.Builder()
        .audioStreamInfo(audioStreamInfo)
        .bufferSizeInBytes(bufferSizeInBytes)
        .deviceId(getSpeaker().getId())
        .isOffload(false)
        .build();
```

1.1.5 创建音频渲染器

```
// 创建音频渲染器
audioRenderer = new AudioRenderer(
    audioRendererInfo, PlayMode.MODE_STREAM);
```

1.2 使用音频渲染器

1.2.1 播放

```
// 播放
audioRenderer.start();

// 异步读取音频文件
getGlobalTaskDispatcher(TaskPriority.DEFAULT)
    .asyncDispatch(() -> readAudioFile(
        openAudioFile("The Destruction of Laputa.wav")));
```

1.2.2 打开并读取外部存储中的音频文件

/storage/emulated/0/Ringtones/tmp/The Destruction of Laputa.wav

```
private FileDescriptor openAudioFile(String filename) {
    FileDescriptor fd = null;

    DataAbilityHelper helper = DataAbilityHelper.creator(this);
    ResultSet result = null;

    try {
        DataAbilityPredicates predicates = new DataAbilityPredicates(
            Media.DISPLAY_NAME + "='" + filename + "'");
        result = helper.query(Audio.Media.EXTERNAL_DATA_ABILITY_URI,
            new String[]{Audio.Media.ID}, predicates);

        if (result.goToFirstRow()) {
            int id = result.getInt(0);
            Uri uri = Uri.appendEncodedPathToUri(
                Audio.Media.EXTERNAL_DATA_ABILITY_URI,
                String.valueOf(id));

            fd = helper.openFile(uri, "r");
        }
    }
    catch (Exception exception) {
        HiLog.info(label, exception.getLocalizedMessage());
    }
    finally {
        if (result != null)
            result.close();
    }

    return fd;
}

private void readAudioFile(FileDescriptor fd) {
    try {
        FileInputStream is = new FileInputStream(fd);

        byte[] buffer = new byte[bufferSizeInBytes];
        int bytes;
        while ((bytes = is.read(buffer)) != -1)
            audioRenderer.write(buffer, 0, bytes);

        is.close();
    }
    catch (Exception exception) {
        HiLog.info(label, exception.getLocalizedMessage());
    }
}
```

1.2.3 暂停

```
// 暂停
audioRenderer.pause();
```

1.2.4 停止

```
// 停止
audioRenderer.stop();
```

1.3 销毁音频渲染器

```
// 销毁音频渲染器
audioRenderer.release();
```

> 例程：AudioRenderer

...\AudioRenderer\entry\src\main\config.json

```
{
  ...
  "module": {
    ...
    "abilities": [
      {
        ...
        "configChanges": [
          "orientation"
        ]
      }
    ]
  }
}
```

...\AudioRenderer\entry\src\main\resources\base\graphic\background_blue_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:shape="oval">
    <solid ohos:color="#00a2e8"/>
</shape>
```

...\AudioRenderer\entry\src\main\resources\base\graphic\background_green_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:shape="oval">
    <solid ohos:color="#22b14c"/>
</shape>
```

...\AudioRenderer\entry\src\main\resources\base\graphic\background_orange_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:shape="oval">
    <solid ohos:color="#ff7f27"/>
</shape>
```

...\AudioRenderer\entry\src\main\resources\base\layout\ability_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<DirectionalLayout
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:height="match_parent"
    ohos:width="match_parent"
    ohos:alignment="center"
    ohos:orientation="vertical"
    ohos:background_element="#804080">

    <Button
        ohos:id="$+id:btnPlay"
        ohos:height="120vp"
        ohos:width="120vp"
        ohos:background_element="$graphic:background_blue_button"
        ohos:text="播放"
        ohos:text_size="28fp"
        ohos:text_color="#ffffff"
        />

    <Button
        ohos:id="$+id:btnPause"
        ohos:height="120vp"
        ohos:width="120vp"
        ohos:top_margin="20vp"
        ohos:background_element="$graphic:background_green_button"
        ohos:text="暂停"
        ohos:text_size="28fp"
        ohos:text_color="#ffffff"
        />

    <Button
        ohos:id="$+id:btnStop"
        ohos:height="120vp"
        ohos:width="120vp"
        ohos:top_margin="20vp"
        ohos:background_element="$graphic:background_orange_button"
        ohos:text="停止"
        ohos:text_size="28fp"
        ohos:text_color="#ffffff"
        />

</DirectionalLayout>
```

...\AudioRenderer\entry\src\main\resources\base\layout\ability_main_landscape.xml

```
<?xml version="1.0" encoding="utf-8"?>
<DirectionalLayout
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:height="match_parent"
    ohos:width="match_parent"
    ohos:alignment="center"
    ohos:orientation="horizontal"
    ohos:background_element="#804080">

    <Button
        ohos:id="$+id:btnPlay"
        ohos:height="120vp"
        ohos:width="120vp"
        ohos:background_element="$graphic:background_blue_button"
        ohos:text="播放"
        ohos:text_size="28fp"
        ohos:text_color="#ffffff"
        />

    <Button
        ohos:id="$+id:btnPause"
        ohos:height="120vp"
        ohos:width="120vp"
        ohos:left_margin="20vp"
        ohos:background_element="$graphic:background_green_button"
        ohos:text="暂停"
        ohos:text_size="28fp"
        ohos:text_color="#ffffff"
        />

    <Button
        ohos:id="$+id:btnStop"
        ohos:height="120vp"
        ohos:width="120vp"
        ohos:left_margin="20vp"
        ohos:background_element="$graphic:background_orange_button"
        ohos:text="停止"
        ohos:text_size="28fp"
        ohos:text_color="#ffffff"
        />

</DirectionalLayout>
```

...\AudioRenderer\entry\src\main\java\com\minwei\audiorenderer\slice\MainAbilitySlice.java

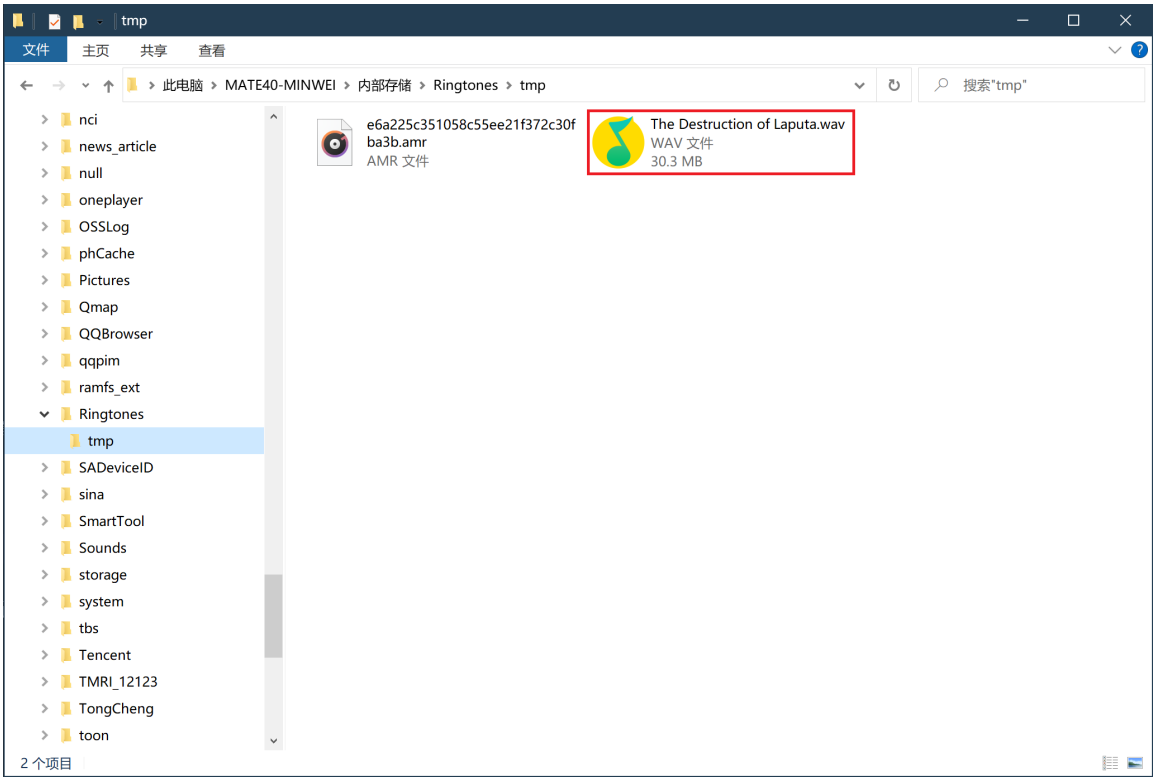
```
public class MainAbilitySlice extends AbilitySlice {
    private static final HiLogLabel label = new HiLogLabel(
        HiLog.LOG_APP, 0x00101,
        MainAbilitySlice.class.getCanonicalName());

    private AudioRenderer audioRenderer; // 音频渲染器
    private int bufferSizeInBytes; // 音频渲染器缓冲区大小

    @Override
    public void onStart(Intent intent) {
        ...
        setButtonListener();
        createAudioRenderer();
    }
    ...
    @Override
    protected void onStop() {
        super.onStop();

        // 停止
        audioRenderer.stop();
        // 销毁音频渲染器
        audioRenderer.release();
    }

    @Override
    protected void onOrientationChanged(
        DisplayOrientation displayOrientation) {
        super.onOrientationChanged(displayOrientation);

        if (displayOrientation == DisplayOrientation.LANDSCAPE)
            super.setUIContent(
                ResourceTable.Layout_ability_main_landscape);
        else
            super.setUIContent(ResourceTable.Layout_ability_main);

        setButtonListener();
    }

    private void setButtonListener() {
        findComponentById(ResourceTable.Id_btnPlay)
            .setClickedListener(component -> play());
        findComponentById(ResourceTable.Id_btnPause)
            .setClickedListener(component -> pause());
        findComponentById(ResourceTable.Id_btnStop)
            .setClickedListener(component -> stop());
    }

    private void createAudioRenderer() {
        // 采样率
        int sampleRate = 44100;
        // 音频流标志
        AudioStreamFlag audioStreamFlag =
            AudioStreamFlag.AUDIO_STREAM_FLAG_DIRECT_OUTPUT;
```

```
        // 编码格式
        EncodingFormat encodingFormat = EncodingFormat.ENCODING_PCM_16BIT;
        // 声道数
        ChannelMask channelMask = ChannelMask.CHANNEL_OUT_STEREO;
        // 使用类型
        StreamUsage streamUsage = StreamUsage.STREAM_USAGE_MEDIA;

        // 音频流信息
        AudioStreamInfo audioStreamInfo =
            new AudioStreamInfo.Builder()
                .sampleRate(sampleRate)
                .audioStreamFlag(audioStreamFlag)
                .encodingFormat(encodingFormat)
                .channelMask(channelMask)
                .streamUsage(streamUsage)
                .build();

        // 获取音频渲染器缓冲区大小
        bufferSizeInBytes = AudioRenderer.getMinBufferSize(
            sampleRate, encodingFormat, channelMask);

        // 音频渲染器信息
        AudioRendererInfo audioRendererInfo =
            new AudioRendererInfo.Builder()
                .audioStreamInfo(audioStreamInfo)
                .bufferSizeInBytes(bufferSizeInBytes)
                .deviceId(getSpeaker().getId())
                .isOffload(false)
                .build();

        // 创建音频渲染器
        audioRenderer = new AudioRenderer(
            audioRendererInfo, PlayMode.MODE_STREAM);
    }

    private void play() {
        // 播放
        audioRenderer.start();

        // 异步读取音频文件
        getGlobalTaskDispatcher(TaskPriority.DEFAULT)
            .asyncDispatch(() -> readAudioFile(
                openAudioFile("The Destruction of Laputa.wav")));
    }

    private void pause() {
        // 暂停
        audioRenderer.pause();
    }

    private void stop() {
        // 停止
        audioRenderer.stop();
    }
```

```
    private AudioDeviceDescriptor getSpeaker() {
        AudioDeviceDescriptor speaker = null;

        // 获取音频输出设备列表
        AudioDeviceDescriptor[] devices =
            AudioManager.getDevices(DeviceFlag.OUTPUT_DEVICES_FLAG);

        // 查找扬声器设备
        for (AudioDeviceDescriptor device : devices)
            if (device.getType() == DeviceType.SPEAKER) {
                speaker = device;
                break;
            }

        return speaker;
    }

    private FileDescriptor openAudioFile(String filename) {
        FileDescriptor fd = null;

        DataAbilityHelper helper = DataAbilityHelper.creator(this);
        ResultSet result = null;

        try {
            /*
            result = helper.query(Media.EXTERNAL_DATA_ABILITY_URI,
                new String[]{Audio.Media.ID, Media.DISPLAY_NAME}, null);
            HiLog.info(label, "找到%{public}d个音频文件",
                result.getRowCount());

            while (result.goToNextRow()) {
                int id = result.getInt(0);
                Uri uri = Uri.appendEncodedPathToUri(
                    Audio.Media.EXTERNAL_DATA_ABILITY_URI,
                    String.valueOf(id));
                String displayName = result.getString(1);
                HiLog.info(label, uri.toString() + " -> " + displayName);
            }
            */
            DataAbilityPredicates predicates = new DataAbilityPredicates(
                Media.DISPLAY_NAME + "='" + filename + "'");
            result = helper.query(Audio.Media.EXTERNAL_DATA_ABILITY_URI,
                new String[]{Audio.Media.ID}, predicates);

            if (result.goToFirstRow()) {
                int id = result.getInt(0);
                Uri uri = Uri.appendEncodedPathToUri(
                    Audio.Media.EXTERNAL_DATA_ABILITY_URI,
                    String.valueOf(id));

                fd = helper.openFile(uri, "r");
            }
        }
        catch (Exception exception) {
            HiLog.info(label, exception.getLocalizedMessage());
```

```
        }
        finally {
            if (result != null)
                result.close();
        }

        return fd;
    }

    private void readAudioFile(FileDescriptor fd) {
        try {
            FileInputStream is = new FileInputStream(fd);

            byte[] buffer = new byte[bufferSizeInBytes];
            int bytes;
            while ((bytes = is.read(buffer)) != -1)
                audioRenderer.write(buffer, 0, bytes);

            is.close();
        }
        catch (Exception exception) {
            HiLog.info(label, exception.getLocalizedMessage());
        }
    }
}
```

将音频文件"The Destruction of Laputa.wav"复制到手机外部存储的"/storage/emulated/0/Ringtones/tmp/"目录下：



运行效果如下图所示：

2 短音播放

2.1 播放系统音

```
SoundPlayer soundPlayer = new SoundPlayer("package");
soundPlayer.playSound(SoundType.KEY_CLICK, 1.0f);
```

2.2 播放双频音

```
SoundPlayer soundPlayer = new SoundPlayer();
soundPlayer.createSound(ToneType.DTMF_0, 1000);
soundPlayer.play();
```

2.3 播放资源音

```
SoundPlayer soundPlayer = new SoundPlayer(
    AudioVolumeType.STREAM_MUSIC.getValue());
int soundId = soundPlayer.createSound(
    this, ResourceTable.Media_nokia);
soundPlayer.setOnCreateCompleteListener(
    new OnCreateCompleteListener() {
        @Override
        public void onCreateComplete(
            SoundPlayer soundPlayer, int soundId, int status) {
            if (status == 0) {
                int taskId = soundPlayer.play(soundId);
                soundPlayer.setPlaySpeedRate(taskId, 1.0f);
                soundPlayer.setVolume(taskId, 1.0f);
                soundPlayer.setLoop(taskId, 0);
            }
        }
    });
```

> 例程：SoundPlayer

...\SoundPlayer\entry\src\main\config.json

```
{
  ...
  "module": {
    ...
    "abilities": [
      {
        ...
        "configChanges": [
          "orientation"
        ]
      }
    ]
  }
}
```

...\SoundPlayer\entry\src\main\resources\base\media\nokia.mp3

nokia.mp3

...\SoundPlayer\entry\src\main\resources\base\graphic\background_blue_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:shape="oval">
    <solid ohos:color="#00a2e8"/>
</shape>
```

...\SoundPlayer\entry\src\main\resources\base\graphic\background_green_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:shape="oval">
    <solid ohos:color="#22b14c"/>
</shape>
```

...\SoundPlayer\entry\src\main\resources\base\graphic\background_orange_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:shape="oval">
    <solid ohos:color="#ff7f27"/>
</shape>
```

...\SoundPlayer\entry\src\main\resources\base\layout\ability_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<DirectionalLayout
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:height="match_parent"
    ohos:width="match_parent"
    ohos:alignment="center"
    ohos:orientation="vertical"
    ohos:background_element="#632523">

    <Button
        ohos:id="$+id:btnSystem"
        ohos:height="120vp"
        ohos:width="120vp"
        ohos:background_element="$graphic:background_blue_button"
        ohos:text="系统音"
        ohos:text_size="28fp"
        ohos:text_color="#ffffff"
        />

    <Button
        ohos:id="$+id:btnTone"
        ohos:height="120vp"
        ohos:width="120vp"
        ohos:top_margin="20vp"
        ohos:background_element="$graphic:background_green_button"
        ohos:text="双频音"
        ohos:text_size="28fp"
        ohos:text_color="#ffffff"
        />

    <Button
        ohos:id="$+id:btnResource"
        ohos:height="120vp"
        ohos:width="120vp"
        ohos:top_margin="20vp"
        ohos:background_element="$graphic:background_orange_button"
        ohos:text="资源音"
        ohos:text_size="28fp"
        ohos:text_color="#ffffff"
        />

</DirectionalLayout>
```

...\SoundPlayer\entry\src\main\resources\base\layout\ability_main_landscape.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<DirectionalLayout
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:height="match_parent"
    ohos:width="match_parent"
    ohos:alignment="center"
    ohos:orientation="horizontal"
    ohos:background_element="#632523">

    <Button
        ohos:id="$+id:btnSystem"
        ohos:height="120vp"
        ohos:width="120vp"
        ohos:background_element="$graphic:background_blue_button"
        ohos:text="系统音"
        ohos:text_size="28fp"
        ohos:text_color="#ffffff"
        />

    <Button
        ohos:id="$+id:btnTone"
        ohos:height="120vp"
        ohos:width="120vp"
        ohos:left_margin="20vp"
        ohos:background_element="$graphic:background_green_button"
        ohos:text="双频音"
        ohos:text_size="28fp"
        ohos:text_color="#ffffff"
        />

    <Button
        ohos:id="$+id:btnResource"
        ohos:height="120vp"
        ohos:width="120vp"
        ohos:left_margin="20vp"
        ohos:background_element="$graphic:background_orange_button"
        ohos:text="资源音"
        ohos:text_size="28fp"
        ohos:text_color="#ffffff"
        />

</DirectionalLayout>
```

...\SoundPlayer\entry\src\main\java\com\minwei\soundplayer\slice\MainAbilitySlice.java

```
public class MainAbilitySlice extends AbilitySlice {
    @Override
    public void onStart(Intent intent) {
        ...
        setButtonListener();
    }
    ...
    @Override
    protected void onOrientationChanged(
        DisplayOrientation displayOrientation) {
        super.onOrientationChanged(displayOrientation);

        if (displayOrientation == DisplayOrientation.LANDSCAPE)
            super.setUIContent(
                ResourceTable.Layout_ability_main_landscape);
        else
            super.setUIContent(ResourceTable.Layout_ability_main);

        setButtonListener();
    }

    private void setButtonListener() {
        findComponentById(ResourceTable.Id_btnSystem)
            .setClickedListener(component -> playSystem());

        findComponentById(ResourceTable.Id_btnTone)
            .setClickedListener(component -> playTone());

        findComponentById(ResourceTable.Id_btnResource)
            .setClickedListener(component -> playResource());
    }

    private void playSystem() {
        SoundPlayer soundPlayer = new SoundPlayer("packageName");
        soundPlayer.playSound(SoundType.KEY_CLICK, 1.0f);
    }

    private void playTone() {
        SoundPlayer soundPlayer = new SoundPlayer();
        soundPlayer.createSound(ToneType.DTMF_0, 1000);
        soundPlayer.play();
    }

    private void playResource() {
        SoundPlayer soundPlayer = new SoundPlayer(
            AudioVolumeType.STREAM_MUSIC.getValue());
        int soundId = soundPlayer.createSound(
            this, ResourceTable.Media_nokia);
        soundPlayer.setOnCreateCompleteListener(
            new OnCreateCompleteListener() {
                @Override
                public void onCreateComplete(
                    SoundPlayer soundPlayer, int soundId, int status) {
                    if (status == 0) {
```

```
                    int taskId = soundPlayer.play(soundId);
                    soundPlayer.setPlaySpeedRate(taskId, 1.0f);
                    soundPlayer.setVolume(taskId, 1.0f);
                    soundPlayer.setLoop(taskId, 0);
                }
            }
        });
    }
}
```

运行效果如下图所示：

3 音频播放

3.1 Player回调

```
public class PlayerCallback implements IPlayerCallback {
    @Override
    public void onPrepared() {
        // 播放开始
    }

    @Override
    public void onMessage(int i, int i1) {
    }

    @Override
    public void onError(int i, int i1) {
        // 发生错误(i：错误类型，i1：错误号)
    }

    @Override
    public void onResolutionChanged(int i, int i1) {
    }

    @Override
    public void onPlayBackComplete() {
        // 播放结束
    }

    @Override
    public void onRewindToComplete() {
    }

    @Override
    public void onBufferingChange(int i) {
    }

    @Override
    public void onNewTimedMetaData(
        Player.MediaTimedMetaData mediaTimedMetaData) {
    }

    @Override
    public void onMediaTimeIncontinuity(
        Player.MediaTimeInfo mediaTimeInfo) {
    }
}
```

3.2 启动播放

```
player = new Player(getContext());
// 播放音频资源
RawFileDescriptor rawFileDescriptor = getResourceManager()
    .getRawFileEntry("resources/rawfile/caiqin.mp3")
    .openRawFileDescriptor();
Source source = new Source(
    rawFileDescriptor.getFileDescriptor(),
    rawFileDescriptor.getStartPosition(),
    rawFileDescriptor.getFileSize());
/* 播放音频文件
FileDescriptor fileDescriptor = new FileInputStream(
    new File(getFilesDir() + "/caiqin.mp3")).getFD();
Source source = new Source(fileDescriptor);
*/
/* 播放音频链接
String URI = "https://www.audio.com/caiqin.mp3";
Source source = new Source(URI);
*/
player.setSource(source);
player.setPlayerCallback(new PlayerCallback(this));
player.prepare();
player.play();
```

3.3 暂停播放

```
player.pause();
```

3.4 终止播放

```
player.stop();
player.release();
player = null;
```

3.5 随机播放

```
player.rewindTo(位置微秒值);
```

3.6 注意事项

3.6.1 进度线程

为了通过滑竿组件和时间文本实时回显播放进度，拟开启独立于UI线程的进度线程跟踪播放进度，该线程调用如下方法：

- player.getDuration()：返回媒体总时长(毫秒)
- player.getCurrentTime()：返回当前播放位置(毫秒)

3.6.2 UI线程

所有关乎界面的更新操作，都应该放到UI线程中执行。

```
getUITaskDispatcher().asyncDispatch(new Runnable() {
    @Override
    public void run() {
        // 更新界面
    }
});
```

3.6.3 滑竿组件

播放器中的滑竿组件既用于输出Player的当前播放位置，又用于输入用户通过拖拽滑块人为指定的播放位置。输出输入需要协调，避免相互干扰和死循环。

3.6.4 屏向感知

默认情况下横竖屏切换会导致页面被重建，如果这时正在播放，则播放过程会被打断，为了避免这种情况的发生，可在config.json文件关于相应Ability的描述中加入如下配置：

```
"configChanges": [
  "orientation"
]
```

> 例程：AudioPlayer

...\AudioPlayer\entry\src\main\config.json

```
{
  ...
  "module": {
    ...
    "abilities": [
      {
        ...
        "configChanges": [
          "orientation"
        ]
      }
    ],
    "reqPermissions": [
      {
        "name": "ohos.permission.INTERNET"
      }
    ]
  }
}
```

...\AudioPlayer\entry\src\main\resources\rawfile\caiqin.mp3

caiqin.mp3

...\AudioPlayer\entry\src\main\resources\base\media\record.png

...\AudioPlayer\entry\src\main\resources\base\graphic\background_red_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:shape="oval">
    <solid ohos:color="#ff4040"/>
</shape>
```

...\AudioPlayer\entry\src\main\resources\base\graphic\background_green_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:shape="oval">
    <solid ohos:color="#22b14c"/>
</shape>
```

...\AudioPlayer\entry\src\main\resources\base\graphic\background_blue_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:shape="oval">
    <solid ohos:color="#00a2e8"/>
</shape>
```

...\AudioPlayer\entry\src\main\resources\base\layout\ability_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<DirectionalLayout
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:height="match_parent"
    ohos:width="match_parent"
    ohos:padding="10vp"
    ohos:background_element="#004080">

    <Image
        ohos:height="match_parent"
        ohos:weight="1"
        ohos:width="match_parent"
        ohos:image_src="$media:record"
        />

    <DirectionalLayout
        ohos:height="match_content"
        ohos:width="match_parent"
        ohos:alignment="center">

        <Text
            ohos:id="$+id:txt"
            ohos:height="match_content"
            ohos:width="match_content"
            ohos:text="00:00:00/00:00:00"
            ohos:text_size="20fp"
            ohos:text_color="#ffffff"
            />

        <Slider
            ohos:id="$+id:sld"
            ohos:height="match_content"
            ohos:width="match_parent"
            ohos:progress_width="5vp"
            ohos:progress_color="#ff7f27"
            />

        <DirectionalLayout
            ohos:height="match_content"
            ohos:width="match_content"
            ohos:orientation="horizontal">

            <Button
                ohos:id="$+id:btnPlay"
                ohos:height="match_content"
                ohos:width="match_content"
                ohos:padding="8vp"
                ohos:background_element="$graphic:background_red_button"
                ohos:text="播放"
                ohos:text_size="20fp"
                ohos:text_color="#ffffff"
                />

            <Button
```

```
                ohos:id="$+id:btnPause"
                ohos:height="match_content"
                ohos:width="match_content"
                ohos:padding="8vp"
                ohos:left_margin="5vp"
                ohos:background_element="$graphic:background_green_button"
                ohos:text="暂停"
                ohos:text_size="20fp"
                ohos:text_color="#ffffff"
                />

            <Button
                ohos:id="$+id:btnStop"
                ohos:height="match_content"
                ohos:width="match_content"
                ohos:padding="8vp"
                ohos:left_margin="5vp"
                ohos:background_element="$graphic:background_blue_button"
                ohos:text="停止"
                ohos:text_size="20fp"
                ohos:text_color="#ffffff"
                />

        </DirectionalLayout>

    </DirectionalLayout>

</DirectionalLayout>
```

...\AudioPlayer\entry\src\main\java\com\minwei\audioplayer\common\ProgressThread.java

```
public class ProgressThread extends Thread {
    private MainAbilitySlice slice;
    private int current = 0, duration;
    private boolean term = false;

    public ProgressThread(MainAbilitySlice slice) {
        this.slice = slice;
    }

    public void terminate() {
        term = true;
    }

    @Override
    public void run() {
        init();

        while (!term && slice.player != null) {
            update();
            try { sleep(1000); }
            catch (InterruptedException exception) {}
        }

        deinit();
    }

    private void init() {
        duration = slice.player.getDuration();

        updateSlider();
        updateText();
    }

    private void update() {
        current = this.slice.player.getCurrentTime();

        if (!slice.dragging)
            updateCurrent();

        updateText();
    }

    private void deinit() {
        if (slice.player != null) {
            slice.player.stop();
            slice.player.release();
            slice.player = null;
        }

        slice.getUITaskDispatcher().asyncDispatch(new Runnable() {
            @Override
            public void run() {
                slice.btnPause.setText("暂停");
                slice.slider.setProgressValue(0);
```

```
                slice.text.setText("00:00:00/00:00:00");
            }
        });
    }

    private void updateCurrent() {
        slice.getUITaskDispatcher().asyncDispatch(new Runnable() {
            @Override
            public void run() {
                slice.slider.setProgressValue(current);
            }
        });
    }

    private void updateSlider() {
        slice.getUITaskDispatcher().asyncDispatch(new Runnable() {
            @Override
            public void run() {
                slice.slider.setMaxValue(duration);
                slice.slider.setProgressValue(current);
            }
        });
    }

    private void updateText() {
        slice.getUITaskDispatcher().asyncDispatch(new Runnable() {
            @Override
            public void run() {
                Calendar calendar = Calendar.getInstance();
                SimpleDateFormat formatter =
                    new SimpleDateFormat("HH:mm:ss");
                formatter.setTimeZone(TimeZone.getTimeZone("GMT"));

                calendar.setTimeInMillis(current);
                String strCurrent = formatter.format(calendar.getTime());

                calendar.setTimeInMillis(duration);
                String strDuration = formatter.format(calendar.getTime());

                slice.text.setText(strCurrent + "/" + strDuration);
            }
        });
    }
}
```

...\AudioPlayer\entry\src\main\java\com\minwei\audioplayer\common\PlayerCallback.java

```
public class PlayerCallback implements IPlayerCallback {
    private static final HiLogLabel label = new HiLogLabel(
        HiLog.LOG_APP, 0x00101, "PlayerCallback");

    private ProgressThread thread;

    public PlayerCallback(MainAbilitySlice slice) {
        thread = new ProgressThread(slice);
    }

    @Override
    public void onPrepared() {
        HiLog.info(label, "onPrepared()");

        thread.start();
    }

    @Override
    public void onMessage(int i, int i1) {
        HiLog.info(label, "onMessage() %{public}d %{public}d", i, i1);
    }

    @Override
    public void onError(int i, int i1) {
        HiLog.info(label, "onError() %{public}d %{public}d", i, i1);
    }

    @Override
    public void onResolutionChanged(int i, int i1) {
        HiLog.info(label,
            "onResolutionChanged() %{public}d %{public}d", i, i1);
    }

    @Override
    public void onPlayBackComplete() {
        HiLog.info(label, "onPlayBackComplete()");

        thread.terminate();
    }

    @Override
    public void onRewindToComplete() {
        HiLog.info(label, "onRewindToComplete()");
    }

    @Override
    public void onBufferingChange(int i) {
        HiLog.info(label, "onBufferingChange() %{public}d", i);
    }

    @Override
    public void onNewTimedMetaData(
        Player.MediaTimedMetaData mediaTimedMetaData) {
        HiLog.info(label, "onNewTimedMetaData()");
```

```
    }

    @Override
    public void onMediaTimeIncontinuity(
        Player.MediaTimeInfo mediaTimeInfo) {
        HiLog.info(label, "onMediaTimeIncontinuity() %{public}d",
            mediaTimeInfo.mediaTimeUs);
    }
}
```

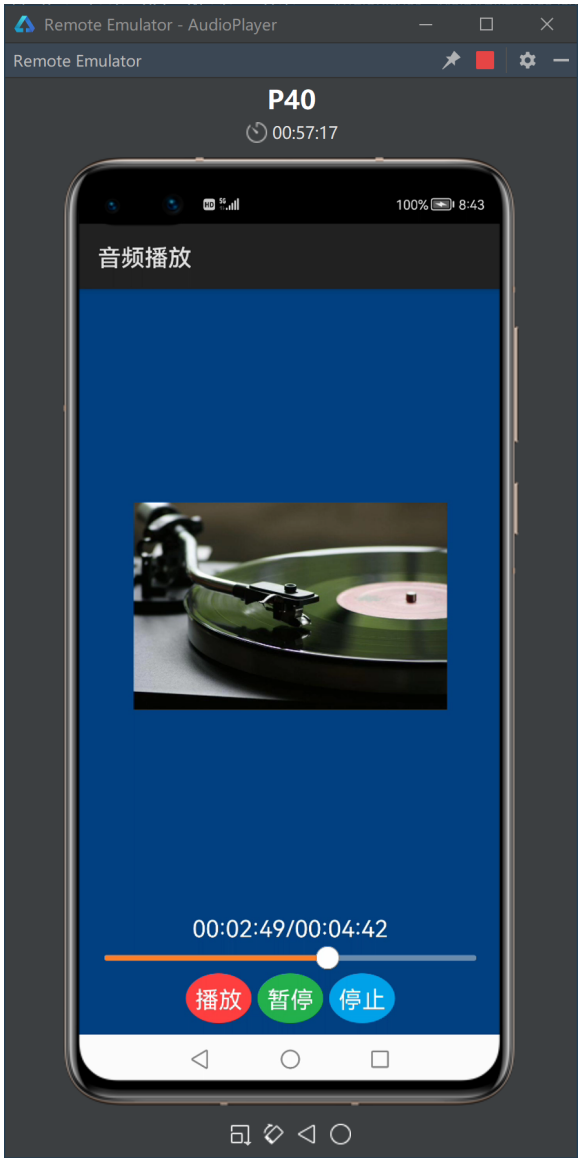
...\AudioPlayer\entry\src\main\java\com\minwei\audioplayer\slice\MainAbilitySlice.java

```
public class MainAbilitySlice extends AbilitySlice {
    private static final HiLogLabel label = new HiLogLabel(
        HiLog.LOG_APP, 0x00101, "MainAbilitySlice");

    public Button btnPlay, btnPause, btnStop;
    public Slider slider;
    public Text text;

    public Player player = null;
    public boolean dragging = false;

    @Override
    public void onStart(Intent intent) {
        ...
        btnPlay = (Button)findComponentById(ResourceTable.Id_btnPlay);
        btnPause = (Button)findComponentById(ResourceTable.Id_btnPause);
        btnStop = (Button)findComponentById(ResourceTable.Id_btnStop);
        slider = (Slider)findComponentById(ResourceTable.Id_sld);
        text = (Text)findComponentById(ResourceTable.Id_txt);

        btnPlay.setClickedListener(component -> {
            if (player == null)
                playResource();
                //playFile();
                //playUrl();
        });

        btnPause.setClickedListener(component -> {
            if (player != null) {
                if (player.isNowPlaying()) {
                    player.pause();

                    getUITaskDispatcher().asyncDispatch(new Runnable() {
                        @Override
                        public void run() {
                            btnPause.setText("继续");
                        }
                    });
                }
                else {
                    player.play();

                    getUITaskDispatcher().asyncDispatch(new Runnable() {
                        @Override
                        public void run() {
                            btnPause.setText("暂停");
                        }
                    });
                }
            }
        });

        btnStop.setClickedListener(component -> {
            if (player != null) {
```

```
                player.stop();
                player.release();
                player = null;
            }
        });

        slider.setValueChangedListener(
            new Slider.ValueChangedListener() {
                @Override
                public void onProgressUpdated(
                    Slider slider, int i, boolean b) {
                    if (player != null && dragging)
                        player.rewindTo(i * 1000);
                }

                @Override
                public void onTouchStart(Slider slider) {
                    dragging = true;
                }

                @Override
                public void onTouchEnd(Slider slider) {
                    dragging = false;
                }
            });
    }
    ...
    @Override
    protected void onStop() {
        if (player != null) {
            player.stop();
            player.release();
            player = null;
        }

        super.onStop();
    }

    private void playResource() {
        try {
            player = new Player(getContext());
            RawFileDescriptor rawFileDescriptor = getResourceManager()
                .getRawFileEntry("resources/rawfile/caiqin.mp3")
                .openRawFileDescriptor();
            Source source = new Source(
                rawFileDescriptor.getFileDescriptor(),
                rawFileDescriptor.getStartPosition(),
                rawFileDescriptor.getFileSize());
            player.setSource(source);
            player.setPlayerCallback(new PlayerCallback(this));
            player.prepare();
            player.play();
        }
        catch (Exception exception) {
            HiLog.info(label, exception.toString());
```

```
        }
    }

    private void playFile() {
        try {
            player = new Player(getContext());
            FileDescriptor fileDescriptor = new FileInputStream(
                new File(getFilesDir() + "/caiqin.mp3")).getFD();
            Source source = new Source(fileDescriptor);
            player.setSource(source);
            player.setPlayerCallback(new PlayerCallback(this));
            player.prepare();
            player.play();
        }
        catch (Exception exception) {
            HiLog.info(label, exception.toString());
        }
    }

    private void playUrl() {
        try {
            player = new Player(getContext());
            String URI = "https://www.audio.com/caiqin.mp3";
            Source source = new Source(URI);
            player.setSource(source);
            player.setPlayerCallback(new PlayerCallback(this));
            player.prepare();
            player.play();
        }
        catch (Exception exception) {
            HiLog.info(label, exception.toString());
        }
    }
}
```

运行效果如下图所示：



4 视频播放

4.1 SurfaceProvider组件

```
<ohos.agp.components.surfaceprovider.SurfaceProvider
    ohos:id="$+id:sp"
    ohos:height="match_parent"
    ohos:weight="1"
    ohos:width="match_parent"
    />
```

```
SurfaceProvider surfaceProvider =
    (SurfaceProvider)findComponentById(ResourceTable.Id_sp);
```

4.2 Surface回调

```
public class SurfaceCallback implements SurfaceOps.Callback {
    @Override
    public void surfaceCreated(SurfaceOps surfaceOps) {
        // 调用Player的setVideoSurface()方法为其设置新Surface
        // 新Surface由surfaceOps参数的getSurface()方法获得
    }

    @Override
    public void surfaceChanged(SurfaceOps surfaceOps,
        int i, int i1, int i2) {
    }

    @Override
    public void surfaceDestroyed(SurfaceOps surfaceOps) {
    }
}
```

```
surfaceProvider.getSurfaceOps().get()
    .addCallback(new SurfaceCallback(this));
```

4.3 Player回调

```
public class PlayerCallback implements IPlayerCallback {
    @Override
    public void onPrepared() {
        // 播放开始
    }

    @Override
    public void onMessage(int i, int i1) {
    }

    @Override
    public void onError(int i, int i1) {
        // 发生错误(i：错误类型，i1：错误号)
    }

    @Override
    public void onResolutionChanged(int i, int i1) {
    }

    @Override
    public void onPlayBackComplete() {
        // 播放结束
    }

    @Override
    public void onRewindToComplete() {
    }

    @Override
    public void onBufferingChange(int i) {
    }

    @Override
    public void onNewTimedMetaData(
        Player.MediaTimedMetaData mediaTimedMetaData) {
    }

    @Override
    public void onMediaTimeIncontinuity(
        Player.MediaTimeInfo mediaTimeInfo) {
    }
}
```

4.4 启动播放

```
player = new Player(getContext());
// 播放视频资源
RawFileDescriptor rawFileDescriptor = getResourceManager()
    .getRawFileEntry("resources/rawfile/wow.mp4")
    .openRawFileDescriptor();
Source source = new Source(
    rawFileDescriptor.getFileDescriptor(),
    rawFileDescriptor.getStartPosition(),
    rawFileDescriptor.getFileSize());
/* 播放视频文件
FileDescriptor fileDescriptor = new FileInputStream(
    new File(getFilesDir() + "/wow.mp4")).getFD();
Source source = new Source(fileDescriptor);
*/
/* 播放视频链接
String URI = "https://www.video.com/wow.mp4";
Source source = new Source(URI);
*/
player.setSource(source);
player.setVideoSurface(
    surfaceProvider.getSurfaceOps().get().getSurface());
player.setPlayerCallback(new PlayerCallback(this));
player.enableScreenOn(true);
player.prepare();
player.play();
```

4.5 暂停播放

```
player.pause();
```

4.6 终止播放

```
player.stop();
player.release();
player = null;
```

4.7 随机播放

```
player.rewindTo(位置微秒值);
```

4.8 注意事项

4.8.1 进度线程

为了通过滑竿组件和时间文本实时回显播放进度，拟开启独立于UI线程的进度线程跟踪播放进度，该线程调用如下方法：

- player.getDuration()：返回媒体总时长(毫秒)
- player.getCurrentTime()：返回当前播放位置(毫秒)

4.8.2 UI线程

所有关乎界面的更新操作，都应该放到UI线程中执行。

```
getUITaskDispatcher().asyncDispatch(new Runnable() {
    @Override
    public void run() {
        // 更新界面
    }
});
```

4.8.3 滑竿组件

播放器中的滑竿组件既用于输出Player的当前播放位置，又用于输入用户通过拖拽滑块人为指定的播放位置。输出输入需要协调，避免相互干扰和死循环。

4.8.4 屏向感知

默认情况下横竖屏切换会导致页面被重建，如果这时正在播放，则播放过程会被打断，为了避免这种情况的发生，可在config.json文件关于相应Ability的描述中加入如下配置：

```
"configChanges": [
  "orientation"
]
```

> 例程：VideoPlayer

...\VideoPlayer\entry\src\main\config.json

```
{
  ...
  "module": {
    ...
    "abilities": [
      {
        ...
        "configChanges": [
          "orientation"
        ]
      }
    ],
    "reqPermissions": [
      {
        "name": "ohos.permission.INTERNET"
      }
    ]
  }
}
```

...\VideoPlayer\entry\src\main\resources\rawfile\wow.mp4

wow.mp4

...\VideoPlayer\entry\src\main\resources\base\graphic\background_red_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:shape="oval">
    <solid ohos:color="#ff4040"/>
</shape>
```

...\VideoPlayer\entry\src\main\resources\base\graphic\background_green_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:shape="oval">
    <solid ohos:color="#22b14c"/>
</shape>
```

...\VideoPlayer\entry\src\main\resources\base\graphic\background_blue_button.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:shape="oval">
    <solid ohos:color="#00a2e8"/>
</shape>
```

...\VideoPlayer\entry\src\main\resources\base\layout\ability_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<DirectionalLayout
    xmlns:ohos="http://schemas.huawei.com/res/ohos"
    ohos:height="match_parent"
    ohos:width="match_parent"
    ohos:padding="10vp"
    ohos:background_element="#000000">

    <DirectionalLayout
        ohos:height="match_parent"
        ohos:weight="1"
        ohos:width="match_parent"
        ohos:alignment="center">

        <ohos.agp.components.surfaceprovider.SurfaceProvider
            ohos:id="$+id:sp"
            ohos:height="180vp"
            ohos:width="320vp"
            />

    </DirectionalLayout>

    <DirectionalLayout
        ohos:height="match_content"
        ohos:width="match_parent"
        ohos:alignment="center">

        <Text
            ohos:id="$+id:txt"
            ohos:height="match_content"
            ohos:width="match_content"
            ohos:text="00:00:00/00:00:00"
            ohos:text_size="20fp"
            ohos:text_color="#ffffff"
            />

        <Slider
            ohos:id="$+id:sld"
            ohos:height="match_content"
            ohos:width="match_parent"
            ohos:progress_width="5vp"
            ohos:progress_color="#ff7f27"
            />

        <DirectionalLayout
            ohos:height="match_content"
            ohos:width="match_content"
            ohos:orientation="horizontal">

            <Button
                ohos:id="$+id:btnPlay"
                ohos:height="match_content"
                ohos:width="match_content"
                ohos:padding="8vp"
```

```
                ohos:background_element="$graphic:background_red_button"
                ohos:text="播放"
                ohos:text_size="20fp"
                ohos:text_color="#ffffff"
                />

            <Button
                ohos:id="$+id:btnPause"
                ohos:height="match_content"
                ohos:width="match_content"
                ohos:padding="8vp"
                ohos:left_margin="5vp"
                ohos:background_element="$graphic:background_green_button"
                ohos:text="暂停"
                ohos:text_size="20fp"
                ohos:text_color="#ffffff"
                />

            <Button
                ohos:id="$+id:btnStop"
                ohos:height="match_content"
                ohos:width="match_content"
                ohos:padding="8vp"
                ohos:left_margin="5vp"
                ohos:background_element="$graphic:background_blue_button"
                ohos:text="停止"
                ohos:text_size="20fp"
                ohos:text_color="#ffffff"
                />

        </DirectionalLayout>

    </DirectionalLayout>

</DirectionalLayout>
```

...\VideoPlayer\entry\src\main\java\com\minwei\videoplayer\common\SurfaceCallback.java

```
public class SurfaceCallback implements Callback {
    private static final HiLogLabel label = new HiLogLabel(
        HiLog.LOG_APP, 0x00101, "SurfaceCallback");

    private MainAbilitySlice slice;

    public SurfaceCallback(MainAbilitySlice slice) {
        this.slice = slice;
    }

    @Override
    public void surfaceCreated(SurfaceOps surfaceOps) {
        HiLog.info(label, "surfaceCreated()");

        if (slice.player != null)
            slice.player.setVideoSurface(surfaceOps.getSurface());
    }

    @Override
    public void surfaceChanged(SurfaceOps surfaceOps,
        int i, int i1, int i2) {
        HiLog.info(label,
            "surfaceChanged() %{public}d %{public}d %{public}d",
            i, i1, i2);
    }

    @Override
    public void surfaceDestroyed(SurfaceOps surfaceOps) {
        HiLog.info(label, "surfaceDestroyed()");
    }
}
```

...\VideoPlayer\entry\src\main\java\com\minwei\videoplayer\common\ProgressThread.java

```
public class ProgressThread extends Thread {
    private MainAbilitySlice slice;
    private int current = 0, duration;
    private boolean term = false;

    public ProgressThread(MainAbilitySlice slice) {
        this.slice = slice;
    }

    public void terminate() {
        term = true;
    }

    @Override
    public void run() {
        init();

        while (!term && slice.player != null) {
            update();
            try { sleep(1000); }
            catch (InterruptedException exception) {}
        }

        deinit();
    }

    private void init() {
        duration = slice.player.getDuration();

        updateSlider();
        updateText();
    }

    private void update() {
        current = this.slice.player.getCurrentTime();

        if (!slice.dragging)
            updateCurrent();

        updateText();
    }

    private void deinit() {
        if (slice.player != null) {
            slice.player.stop();
            slice.player.release();
            slice.player = null;
        }

        slice.getUITaskDispatcher().asyncDispatch(new Runnable() {
            @Override
            public void run() {
                slice.surfaceProvider.removeFromWindow();
                slice.btnPause.setText("暂停");
```

```
                slice.slider.setProgressValue(0);
                slice.text.setText("00:00:00/00:00:00");
            }
        });
    }

    private void updateCurrent() {
        slice.getUITaskDispatcher().asyncDispatch(new Runnable() {
            @Override
            public void run() {
                slice.slider.setProgressValue(current);
            }
        });
    }

    private void updateSlider() {
        slice.getUITaskDispatcher().asyncDispatch(new Runnable() {
            @Override
            public void run() {
                slice.slider.setMaxValue(duration);
                slice.slider.setProgressValue(current);
            }
        });
    }

    private void updateText() {
        slice.getUITaskDispatcher().asyncDispatch(new Runnable() {
            @Override
            public void run() {
                Calendar calendar = Calendar.getInstance();
                SimpleDateFormat formatter =
                    new SimpleDateFormat("HH:mm:ss");
                formatter.setTimeZone(TimeZone.getTimeZone("GMT"));

                calendar.setTimeInMillis(current);
                String strCurrent = formatter.format(calendar.getTime());

                calendar.setTimeInMillis(duration);
                String strDuration = formatter.format(calendar.getTime());

                slice.text.setText(strCurrent + "/" + strDuration);
            }
        });
    }
}
```

...\VideoPlayer\entry\src\main\java\com\minwei\videoplayer\common\PlayerCallback.java

```
public class PlayerCallback implements IPlayerCallback {
    private static final HiLogLabel label = new HiLogLabel(
        HiLog.LOG_APP, 0x00101, "PlayerCallback");

    private ProgressThread thread;

    public PlayerCallback(MainAbilitySlice slice) {
        thread = new ProgressThread(slice);
    }

    @Override
    public void onPrepared() {
        HiLog.info(label, "onPrepared()");

        thread.start();
    }

    @Override
    public void onMessage(int i, int i1) {
        HiLog.info(label, "onMessage() %{public}d %{public}d", i, i1);
    }

    @Override
    public void onError(int i, int i1) {
        HiLog.info(label, "onError() %{public}d %{public}d", i, i1);
    }

    @Override
    public void onResolutionChanged(int i, int i1) {
        HiLog.info(label,
            "onResolutionChanged() %{public}d %{public}d", i, i1);
    }

    @Override
    public void onPlayBackComplete() {
        HiLog.info(label, "onPlayBackComplete()");

        thread.terminate();
    }

    @Override
    public void onRewindToComplete() {
        HiLog.info(label, "onRewindToComplete()");
    }

    @Override
    public void onBufferingChange(int i) {
        HiLog.info(label, "onBufferingChange() %{public}d", i);
    }

    @Override
    public void onNewTimedMetaData(
        Player.MediaTimedMetaData mediaTimedMetaData) {
        HiLog.info(label, "onNewTimedMetaData()");
```

```
    }

    @Override
    public void onMediaTimeIncontinuity(
        Player.MediaTimeInfo mediaTimeInfo) {
        HiLog.info(label, "onMediaTimeIncontinuity() %{public}d",
            mediaTimeInfo.mediaTimeUs);
    }
}
```

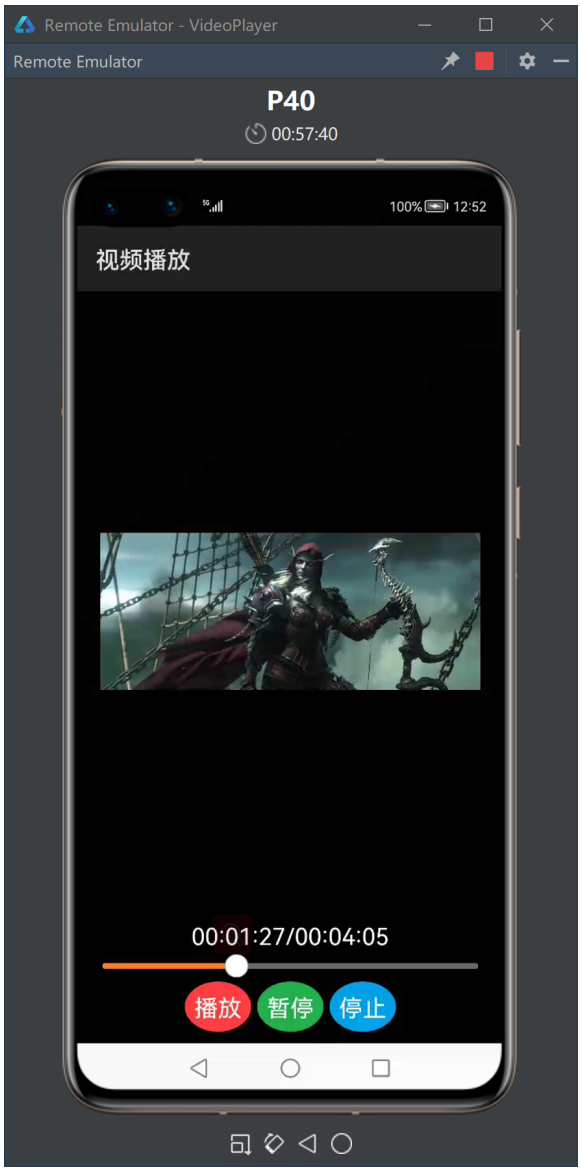
...\VideoPlayer\entry\src\main\java\com\minwei\videoplayer\slice\MainAbilitySlice.java

```
public class MainAbilitySlice extends AbilitySlice {
    private static final HiLogLabel label = new HiLogLabel(
        HiLog.LOG_APP, 0x00101, "MainAbilitySlice");

    public SurfaceProvider surfaceProvider;
    public Button btnPlay, btnPause, btnStop;
    public Slider slider;
    public Text text;

    public Player player = null;
    public boolean dragging = false;

    @Override
    public void onStart(Intent intent) {
        ...
        surfaceProvider = (SurfaceProvider)findComponentById(
            ResourceTable.Id_sp);
        btnPlay = (Button)findComponentById(ResourceTable.Id_btnPlay);
        btnPause = (Button)findComponentById(ResourceTable.Id_btnPause);
        btnStop = (Button)findComponentById(ResourceTable.Id_btnStop);
        slider = (Slider)findComponentById(ResourceTable.Id_sld);
        text = (Text)findComponentById(ResourceTable.Id_txt);

        surfaceProvider.getSurfaceOps().get()
            .addCallback(new SurfaceCallback(this));
        //surfaceProvider.pinToZTop(true);
        surfaceProvider.pinToZTop(false);
        WindowManager.getInstance()
            .getTopWindow().get().setTransparent(true);

        btnPlay.setClickedListener(component -> {
            if (surfaceProvider.getSurfaceOps().isPresent() &&
                player == null)
                playResource();
                //playFile();
                //playUrl();
        });

        btnPause.setClickedListener(component -> {
            if (player != null) {
                if (player.isNowPlaying()) {
                    player.pause();

                    getUITaskDispatcher().asyncDispatch(new Runnable() {
                        @Override
                        public void run() {
                            btnPause.setText("继续");
                        }
                    });
                }
                else {
                    player.play();

                    getUITaskDispatcher().asyncDispatch(new Runnable() {
```

```
                    @Override
                    public void run() {
                        btnPause.setText("暂停");
                    }
                });
            }
        }
    });

    btnStop.setClickedListener(component -> {
        if (player != null) {
            player.stop();
            player.release();
            player = null;
        }
    });

    slider.setValueChangedListener(
        new Slider.ValueChangedListener() {
            @Override
            public void onProgressUpdated(
                Slider slider, int i, boolean b) {
                if (player != null && dragging)
                    player.rewindTo(i * 1000);
            }

            @Override
            public void onTouchStart(Slider slider) {
                dragging = true;
            }

            @Override
            public void onTouchEnd(Slider slider) {
                dragging = false;
            }
        });
}
...
@Override
protected void onStop() {
    if (player != null) {
        player.stop();
        player.release();
        player = null;
    }

    super.onStop();
}

private void playResource() {
    try {
        player = new Player(getContext());
        RawFileDescriptor rawFileDescriptor = getResourceManager()
            .getRawFileEntry("resources/rawfile/wow.mp4")
            .openRawFileDescriptor();
```

```
            Source source = new Source(
                rawFileDescriptor.getFileDescriptor(),
                rawFileDescriptor.getStartPosition(),
                rawFileDescriptor.getFileSize());
            player.setSource(source);
            player.setVideoSurface(
                surfaceProvider.getSurfaceOps().get().getSurface());
            player.setPlayerCallback(new PlayerCallback(this));
            player.enableScreenOn(true);
            player.prepare();
            player.play();
        }
        catch (Exception exception) {
            HiLog.info(label, exception.toString());
        }
    }

    private void playFile() {
        try {
            player = new Player(getContext());
            FileDescriptor fileDescriptor = new FileInputStream(
                new File(getFilesDir() + "/wow.mp4")).getFD();
            Source source = new Source(fileDescriptor);
            player.setSource(source);
            player.setVideoSurface(
                surfaceProvider.getSurfaceOps().get().getSurface());
            player.setPlayerCallback(new PlayerCallback(this));
            player.enableScreenOn(true);
            player.prepare();
            player.play();
        }
        catch (Exception exception) {
            HiLog.info(label, exception.toString());
        }
    }

    private void playUrl() {
        try {
            player = new Player(getContext());
            String URI = "https://www.video.com/wow.mp4";
            Source source = new Source(URI);
            player.setSource(source);
            player.setVideoSurface(
                surfaceProvider.getSurfaceOps().get().getSurface());
            player.setPlayerCallback(new PlayerCallback(this));
            player.enableScreenOn(true);
            player.prepare();
            player.play();
        }
        catch (Exception exception) {
            HiLog.info(label, exception.toString());
        }
    }
}
```

运行效果如下图所示：



更多精彩，敬请期待……

达内集团C++教学部
2021年9月23日