1.UIClasses主要负责UI部分，包含News/Person等

2.AzeroConfig:主要是azero的一些配置文件

3.SaiResource是存放一些资源文件

4.PcmPlayManager是pcm播放器的管理者文件

5.RecordManager负责录制音频

6.AzeroSubclass是AzeroSDK的配置文件

7.MP3ToPcm是负责MP3转pcm

8.inputs是一些本地音频文件

SaiAzeroManager是负责AzeroSDK的管理者。管理着SDK的初始化、唤醒与非唤醒以及音频资源的下发等。

azeroManagerStartRecord调用此API来开启录音

azeroManagerEndRecord调用此API来结束录音

azeroSetUpAzeroSDK调用此API来初始化SDK

azeroManagerWriteData调用此API来为SDK传输录音的数据

azeroManagerReadDataWith: andSize:调用此API来读取SDK返回的TTS数据

azeroManagerReadDataComplete调用此API来判断读取的TTS数据是否完成

azeroManagerReportTtsPlayStateStart用于TTS播放器上报当前播放器的播放状态-Start

azeroManagerReportMp3PlayStateStart用于MP3播放器上报当前播放器的播放状态-Start

azeroManagerReportTtsPlayStateStop用于TTS播放器上报当前播放器的播放状态-Stop

azeroManagerReportMp3PlayStateStop用于MP3播放器上报当前播放器的播放状态-Stop

azeroManagerReportTtsPlayStateFinished用于TTS播放器上报当前播放器的播放状态-Finished

azeroManagerReportMp3PlayStateFinished用于MP3播放器上报当前播放器的播放状态-Finished

azeroManagerReportMp3PlayStateError用于MP3播放器上报当前播放器的播放状态-Error

azeroManagerSentTxet调用此API来发送文字信息到SDK

azeroPlayTtsStatePrepare当SDK准备调用TTS播放器播放TTS数据的时候回调此block

azeroPlaySongUrl:SDK下发将要播放的歌曲url的时候回调此block

azeroListInfo:调用此API来回调SDK返回的列表信息

azeroManagerRenderTemplate:调用此API来回调SDK下发的模板信息

azeroManagerExpress:调用此API来回调SDK下发的表述信息

azeroManagerVadStart:当触发VadStart的回调此函数

azeroManagerVadStop:用于监听vad的结束

azeroLogoutAzeroSDK退出SDK

azeroAddContactsBegin通知SDK开始上传通讯录

azeroAddContactsEnd通知SDK上传通讯录结束

azeroAddContact:调用此API来上传通讯录

azeroQueryContact调用此API来查询通讯录信息

azeroSDKConnectionStatusChangedWithStatus:调用此API来检测SDK的链接状态

azeroManagerSetSongCycleMode:调用此API来设置播放器的播放模式

azeroManagerRemoveAlert:调用此API来移除单个闹钟

azeroManagerRemoveAllAlert:调用此API来移除所有的闹钟

azeroManagerSwitchMode: andValue:切换SDK的模式

azeroButtonPressed:按钮的点击事件

azeroManagerReportAlertPlayStateFinished上报闹钟播放完毕的状态

azeroReportSystemCurrentVolume:向SDK上报系统音量

azeroUpdateMessageWithLevel: tag: messmage:上传日志信息

sdk初始化的逻辑：

初始化AzeroEngine的实例化对象，当需要使用相关功能的时候，用AzeroEngine的实例化对象对其进行注册：例如：

bRet = [engine registerPlatformInterface:self.audioPlayer];

assert(bRet);

XBEchoCancellation是管理录音的类，初始化的重要参数：

_rate = 16000;

_bit = 16;

_channel = 1;

_echoCancellationStatus = XBEchoCancellationStatus_close;

CheckError(AudioUnitSetProperty(myStruct.remoteIOUnit,

kAUVoiceIOProperty_BypassVoiceProcessing,

kAudioUnitScope_Global,

0,

&newEchoCancellationStatus,

sizeof(newEchoCancellationStatus)),

"AudioUnitSetProperty kAUVoiceIOProperty_BypassVoiceProcessing failed");

_echoCancellationStatus = newEchoCancellationStatus == 0 ? XBEchoCancellationStatus_open : XBEchoCancellationStatus_close;设置回声消除的相关参数

SaiMp3ToPcmManager是一个管理MP3转化为pcm的管理者

使用的是lame库来进行转化，其中的一些重要参数的设置：

self.l = hip_decode_init();

lame_set_in_samplerate(self.lame, 16000);

lame_set_decode_only(self.lame, 1);

lame_set_VBR(self.lame, vbr_default);

lame_set_mode(self.lame, MONO);

lame_init_params(self.lame);

AudioQueuePlay是用来播放pcm音频的管理者

初始化的一些重要参数

_audioDescription.mSampleRate = 16000;//采样率

_audioDescription.mFormatID = kAudioFormatLinearPCM;

// 下面这个是保存音频数据的方式的说明，如可以根据大端字节序或小端字节序，浮点数或整数以及不同体位去保存数据

_audioDescription.mFormatFlags = kAudioFormatFlagIsSignedInteger | kAudioFormatFlagIsPacked;

//1单声道 2双声道

// _audioDescription.mFormatFlags = kLinearPCMFormatFlagIsSignedInteger | kAudioFormatFlagsNativeEndian | kAudioFormatFlagIsPacked;

_audioDescription.mChannelsPerFrame = 1;

//每一个packet一侦数据,每个数据包下的桢数，即每个数据包里面有多少桢

_audioDescription.mFramesPerPacket = 1;

//每个采样点16bit量化 语音每采样点占用位数

_audioDescription.mBitsPerChannel = 16;

*audioDescription.mBytesPerFrame = (*audioDescription.mBitsPerChannel / 8) * _audioDescription.mChannelsPerFrame;

//每个数据包的bytes总数，每桢的bytes数*每个数据包的桢数

_audioDescription.mBytesPerPacket = _audioDescription.mBytesPerFrame * _audioDescription.mFramesPerPacket;

AudioQueuePlay主要是使用音频队列来完成的，此队列的主要步骤如下

1、准备播放的音频队列，为每个音频队列缓冲区（Buffer）进行数据填充；

2、当启用AudioQueueStart时，即刻进行播放数据；

3、将队列里第一个缓冲的buffer发送到音频输出区

4、播放队列进入循环模式，音频队列可以进行下一个的音频缓冲区播放

5、回调告诉上层缓冲的buffer已被使用了，然后可以进行下一次的缓冲

6、待上一个已被播放了的音频buffer释放后再次填充buffer

[[SaiAzeroManager sharedAzeroManager] saiAzeroManagerRenderTemplate:^(NSString *renderTemplateStr) {

TYLog(@"hello - saiAzeroManagerRenderTemplate： renderTemplate 内容：%@",renderTemplateStr);

dispatch_async(dispatch_get_main_queue(), ^{

NSDictionary *diction=[SaiJsonConversionModel dictionaryWithJsonString:renderTemplateStr];

if ([diction[@"type"] isEqualToString:@"HelloWeatherTemplate"]) {

[SaiAzeroManager sharedAzeroManager].helloWeatherTemplate=renderTemplateStr;

[SaiNotificationCenter postNotificationName:@"HelloWeatherTemplate" object:**nil** userInfo:@{@"HelloWeatherTemplate":renderTemplateStr}];

return ;

}**else if** ([diction[@"type"] isEqualToString:@"SkillListTipsTemplate"]) {

[SaiNotificationCenter postNotificationName:@"HelloWeatherTemplate" object:**nil** userInfo:@{@"HelloWeatherTemplate":renderTemplateStr}];

return ;

}**else if**([diction[@"type"] isEqualToString:@"ToastTemplate"]){

[MessageAlertView showHudMessage:diction[@"toast"]];

return;

}**else if** ([diction[@"type"] isEqualToString:@"TranslateTemplate"]){

if ([diction[@"value"] isEqualToString:@"start"]) {

}

return;

}

TemplateTypeENUM templateTypet=[QKUITools returnTemplateFromRenderTemplateStr:diction[@"type"]];

switch (templateTypet) {

case RenderPlayerInfo:

```
case EnglishTemplate:

{

NSString *audioItemId = diction[@"audioItemId"];

NSString *defaultaudioItemId = [[NSUserDefaults standardUserDefaults]
objectForKey:@"defaultAudioItemId"];

if ([defaultaudioItemId isEqualToString:audioItemId]||([QKUITools
isBlankArray:diction[@"contents"]]&&![diction containsObjectForKey:@"content"])) {

return ;

}else{

[[NSUserDefaults standardUserDefaults]setValue:audioItemId forKey:@"defaultAudioItemId"];

}

}

break;

case ASMRRenderPlayerInfo:{

NSString *audioItemId = diction[@"audioItemId"];

NSString *defaultaudioItemId = [[NSUserDefaults standardUserDefaults]
objectForKey:@"defaultAudioItemId"];

if ([defaultaudioItemId isEqualToString:audioItemId]||([QKUITools
isBlankArray:diction[@"contents"]]&&![diction containsObjectForKey:@"content"])) {

return ;

}else{

[[NSUserDefaults standardUserDefaults]setValue:audioItemId forKey:@"defaultAudioItemId"];

}

}

break;

case AlertRingtoneTemplate:{

[SaiAzeroManager sharedAzeroManager].alert_token = diction[@"alert_token"];

}

break;

default:{

}

break;
```

```
};

TYLog(@"获取的球体的信息是 %@",renderTemplateStr);

[SaiAzeroManager sharedAzeroManager].renderTemplateStr=renderTemplateStr;

[SaiAzeroManager sharedAzeroManager].songListStr=renderTemplateStr;

if (![[NSUserDefaults standardUserDefaults] boolForKey:@"isFirst"]&&![[NSString stringWithFormat:@"%@",diction[@"type"] ]isEqualToString:@"SphereTemplate"]&&![[NSString stringWithFormat:@"%@",diction[@"type"] ]isEqualToString:@"LauncherTemplate1"]) {

[[SaiHomePageViewController sharedInstance].titleLabel setText:@"可以对我说“返回首页”"];

}

if (weakSelf.responseRenderTemplateStr) {

weakSelf.responseRenderTemplateStr(renderTemplateStr);

}

});

}];
```

通过saiAzeroManagerRenderTemplateblock函数回调一些数据信息，例如首页的球体信息，翻译、技能模板信息等。根据type字段来判断返回的数据类型信息。然后将数据data展示到app的UI中。

```
[[SaiAzeroManager sharedAzeroManager] saiAzeroSongListInfo:^(NSString *songListStr) {

TYLog(@"hello - saiAzeroSongListInfo： renderPlayerInfo 内容：%@",songListStr);

dispatch_async(dispatch_get_main_queue(), ^{

NSDictionary *diction=[SaiJsonConversionModel dictionaryWithJsonString:songListStr];

NSArray *array = diction[@"controls"];

NSDictionary *dic = array[0];

NSString *type = dic[@"type"];

if ([type isEqualToString:@"TOGGLE"]) {

NSString *name = dic[@"name"];

if ([name isEqualToString:@"SINGLE_LOOP"]) {

[SaiAzeroManager sharedAzeroManager].songMode = SongCycleModeSingle;

}else if ([name isEqualToString:@"SHUFFLE"]){

[SaiAzeroManager sharedAzeroManager].songMode = SongCycleModeRandom;

}else if ([name isEqualToString:@"LOOP"]){

[SaiAzeroManager sharedAzeroManager].songMode = SongCycleModeOrder;

}
```

```
}

TemplateTypeENUM templateTypet=[QKUITools
returnTemplateFromRenderTemplateStr:diction[@"type"]];

switch (templateTypet) {

case RenderPlayerInfo:

case EnglishTemplate:{

NSString *audioItemId = diction[@"audioItemId"];

NSString *defaultaudioItemId = [[NSUserDefaults standardUserDefaults]
objectForKey:@"defaultAudioItemId"];

if ([defaultaudioItemId isEqualToString:audioItemId]||([QKUITools
isBlankArray:diction[@"contents"]]&&![diction containsObjectForKey:@"content"])) {

return ;

}else{

[[NSUserDefaults standardUserDefaults]setValue:audioItemId forKey:@"defaultAudioItemId"];

}

}

break;

case ASMRRenderPlayerInfo:{

NSString *audioItemId = diction[@"audioItemId"];

NSString *defaultaudioItemId = [[NSUserDefaults standardUserDefaults]
objectForKey:@"defaultAudioItemId"];

if ([defaultaudioItemId isEqualToString:audioItemId]||([QKUITools
isBlankArray:diction[@"contents"]]&&![diction containsObjectForKey:@"content"])) {

return ;

}else{

[[NSUserDefaults standardUserDefaults]setValue:audioItemId forKey:@"defaultAudioItemId"];

}

}

break;

default:{

}

break;

};
```

```
TYLog(@"获取的球体的信息是 %@",songListStr);

[SaiAzeroManager sharedAzeroManager].songListStr=songListStr;

if (![[NSUserDefaults standardUserDefaults] boolForKey:@"isFirst"]&&![[NSString stringWithFormat:@"%@",diction[@"type"] ]isEqualToString:@"SphereTemplate"]&&![[NSString stringWithFormat:@"%@",diction[@"type"] ]isEqualToString:@"LauncherTemplate1"]) {

[[SaiHomePageViewController sharedInstance].titleLabel setText:@"可以对我说“返回首页”"];

}

if (weakSelf.responseRenderTemplateStr) {

weakSelf.responseRenderTemplateStr(songListStr);

}

});

}];
```

通过saiAzeroSongListInfo函数回调可以返回一些歌曲信息，例如当前播放的歌曲列表信息。还会返回一些歌曲的播放模式信息等。然后根据返回的数据信息进行相应的数据展示。

```
[[SaiAzeroManager sharedAzeroManager] saiAzeroManagerExpress:^(NSString *type, NSString *content) {

TYLog(@"&&&&&&type : %@ ,content : %@",type,content);

TYLog(@"hello - saiAzeroManagerExpress： handleExpressDirectiveFor 内容： %@ %@",type,content);

dispatch_async(dispatch_get_main_queue(), ^{

NSDictionary *dic = [SaiJsonConversionModel dictionaryWithJsonString:content];

NSString *action = dic[@"action"];

if ([action isEqualToString:@"pause"]) {

[[NSUserDefaults standardUserDefaults] setBool:YES forKey:@"kIsPause"];

}else if ([action isEqualToString:@"finish"]){

[[NSUserDefaults standardUserDefaults] setBool:NO forKey:@"kIsPause"];

[[NSUserDefaults standardUserDefaults] setBool:NO forKey:@"kIsStart"];

[[ZXCTimer shareInstance]removeCycleTask:SaiContext.timerQueueItem];

[[ZXCTimer shareInstance]removeCycleTask:SaiContext.queueItem];

NSDictionary *runFeedback=dic[@"runFeedback"];

if (SaiContext.timerQueueItem) {

UIView *bgV = [[UIView alloc] initWithFrame:[UIScreen mainScreen].bounds];

bgV.backgroundColor = [[UIColor blackColor] colorWithAlphaComponent:0.5];
```

```
bgV.backgroundColor = UIColor.redColor;

[[[UIApplication sharedApplication].delegate window] addSubview:bgV];

// UITapGestureRecognizer *tap=[[UITapGestureRecognizer alloc]initWithActionBlock:^(id
_Nonnull sender) {

// [bgV removeFromSuperview];

// }];

// [bgV addGestureRecognizer:tap];

if ([QKUITools isBlankDictionary:runFeedback]) {

return ;

}

SaiRunAlertView *shareView = [[SaiRunAlertView alloc] initAlertView: [NSString
stringWithFormat:@"%@",runFeedback[@"distance"]] time: [NSString
stringWithFormat:@"%d",SaiContext.currentTime] calories:[NSString
stringWithFormat:@"%@",runFeedback[@"calorie"]]];

[shareView setBackblock:^{

[bgV removeFromSuperview];

}];

[bgV addSubview:shareView];

[shareView mas_makeConstraints:^(MASConstraintMaker *make) {

make.edges.equalTo(bgV);

}];

}

SaiContext.currentTime=0;

SaiContext.timerQueueItem=nil;

SaiContext.queueItem=nil;

}

else if ([action isEqualToString:@"resume"]||[action isEqualToString:@"begin"]){

[[NSUserDefaults standardUserDefaults] setBool:NO forKey:@"kIsPause"];

}

else if ([action isEqualToString:@"QUERY_STEP_COUNT"]){

SaiContext.walkDiction=dic;

}
```

```
if (weakSelf.GetDataBlock) {

weakSelf.GetDataBlock(type, content);

}

if ([action isEqualToString:@"goHome"]) {

kWYPlayerVC.isNotSendBack=YES;

[self backAction];

if (![[NSUserDefaults standardUserDefaults] boolForKey:@"isFirst"]) {

[[SaiHomePageViewController sharedInstance] removeView];

}

[KSaiJXHomePageViewController switchIndex:1];

[[SaiHomePageViewController sharedInstance]assignmentAzeroManagerBlockHandle];

}

if ([action isEqualToString:@"Exit"]) {

switch ([SaiAzeroManager sharedAzeroManager].loctionModeType) {

case ModeTranslate:

[[SaiAzeroManager sharedAzeroManager] saiAzeroManagerSwitchMode:ModeHeadset
andValue:YES];

break;

default:

break;

}

[self backAction];

[[SaiAzeroManager sharedAzeroManager] saiAzeroManagerSentTxet:@"拉取场景tj数据"];

[[SaiHomePageViewController sharedInstance]assignmentAzeroManagerBlockHandle];

[[SaiAzeroManager sharedAzeroManager] saiAzeroButtonPressed:ButtonTypePAUSE];

[KSaiJXHomePageViewController switchIndex:1];

}

if ([type isEqualToString:@"ASRText"]) {

[[SaiAzeroManager sharedAzeroManager] saiAzeroManagerRemoveAlert:[SaiAzeroManager
sharedAzeroManager].alert_token];

NSDictionary *contentDictionary=[SaiJsonConversionModel dictionaryWithJsonString:content];

bool finished=[[NSString stringWithFormat:@"%@",contentDictionary[@"finished"]] boolValue];
```

```
if (![QKUITools isBlankString:contentDictionary[@"text"]]) {

[SaiSoundWaveView showMessage:contentDictionary[@"text"]];

}

if (finished) {

dispatch_after(dispatch_time(DISPATCH_TIME_NOW, (int64_t)(0.2 * NSEC_PER_SEC)),
dispatch_get_main_queue(), ^{

[SaiSoundWaveView showMessage:@""];

});

}

}

if ([type isEqualToString:@"LocaleMode"]) {

[[SaiAzeroManager sharedAzeroManager] saiUpdateMessageWithLevel:SaiERROR tag:nil
messmage:[NSString stringWithFormat:@"saiAzeroManagerSwitchMode:ModeHeadset
********* %@",dic]];

NSString *mode = dic[@"mode"];

NSString *value = dic[@"value"];

if ([mode isEqualToString:@"headset"]) {

if ([value isEqualToString:@"ON"]) {

[SaiAzeroManager sharedAzeroManager].loctionModeType = ModeHeadset;

// [SaiNotificationCenter postNotificationName:SaiSetHeadsetModeSuccess object:nil];

}

}else if ([mode isEqualToString:@"translate"]){

if ([value isEqualToString:@"ON"]) {

[SaiAzeroManager sharedAzeroManager].loctionModeType = ModeTranslate;

// [SaiNotificationCenter postNotificationName:SaiSetTranslateModeSuccess object:nil];

}

}

}

});

}];
```

通过saiAzeroManagerExpress函数来返回ASRText数据，以及当前的本地模式，以及一些状态信息等。

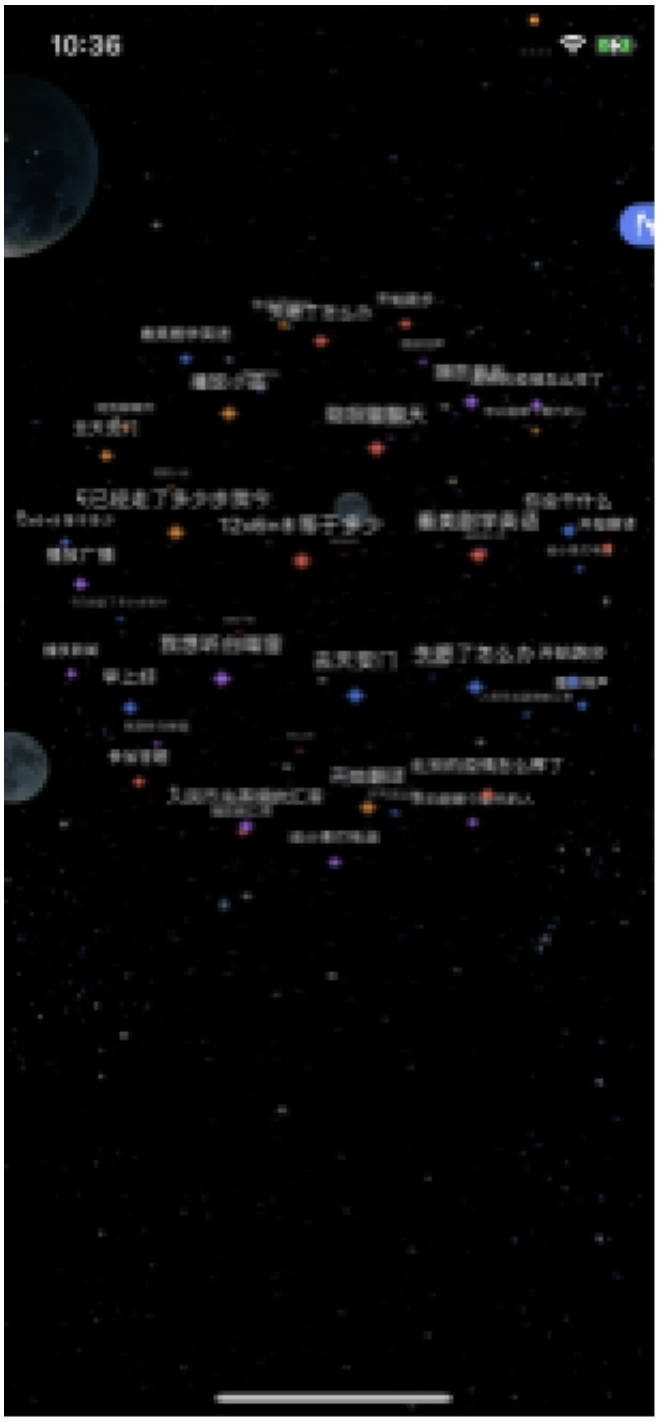
```
[[SaiAzeroManager sharedAzeroManager] saiAzeroManagerVadStart:^{
TYLog(@"
================================================================================
======================== saiAzeroManagerVadStart");
dispatch_async(dispatch_get_main_queue(), ^{
[SaiSoundWaveView showHudAni];
// [SaiSoundWaveView showMessage:@""];
});
}];
[[SaiAzeroManager sharedAzeroManager] saiAzeroManagerVadStop:^{
TYLog(@"
================================================================================
======================== saiAzeroManagerVadEnd");
TYLog(@"--**localDetectorEventSpeechStopDetected3");
dispatch_async(dispatch_get_main_queue(), ^{
[SaiSoundWaveView dismissHudAni];
TYLog(@"--**localDetectorEventSpeechStopDetected4");
});
}];
```

通过saiAzeroManagerVadStart与saiAzeroManagerVadStop来检测Vad的开始与结束。根据开始与结束的回调来进行appUI的相关展示。

```
- (bool)saiAzeroManagerReadDataComplete;
- (void)saiAzeroManagerReportMp3PlayStateStart;
- (void)saiAzeroManagerReportTtsPlayStateFinished;
- (void)saiAzeroManagerReportMp3PlayStateStop;
- (void)saiAzeroManagerReportMp3PlayStateFinished;
- (void)saiAzeroManagerReportAlertPlayStateFinished;
- (void)saiAzeroManagerReportMp3PlayStateError;
```

以上是几个上报状态函数，每当端上的应用发生播放状态的改变时候，都要将端上的播放状态同步到SDK中，此时要调用以上各个函数来上报当前的播放状态，实现SDK与

下图为TA来了APP的首页截图



首页为一个旋转的球体功能展示，上面显示了TA来了的功能。

用户可以按照功能上的文字展示来语音询问控制APP。例如：用户可以按照球体上的文字信息语音："今天的天气怎么样"，APP收到语音指令后，会进行TTS的相关播报，并跳转相应的界面展示。回答完用户的相关问题之后，回跳转回首页界面。并且首页的下方会展示，app识别到的语音文字信息。

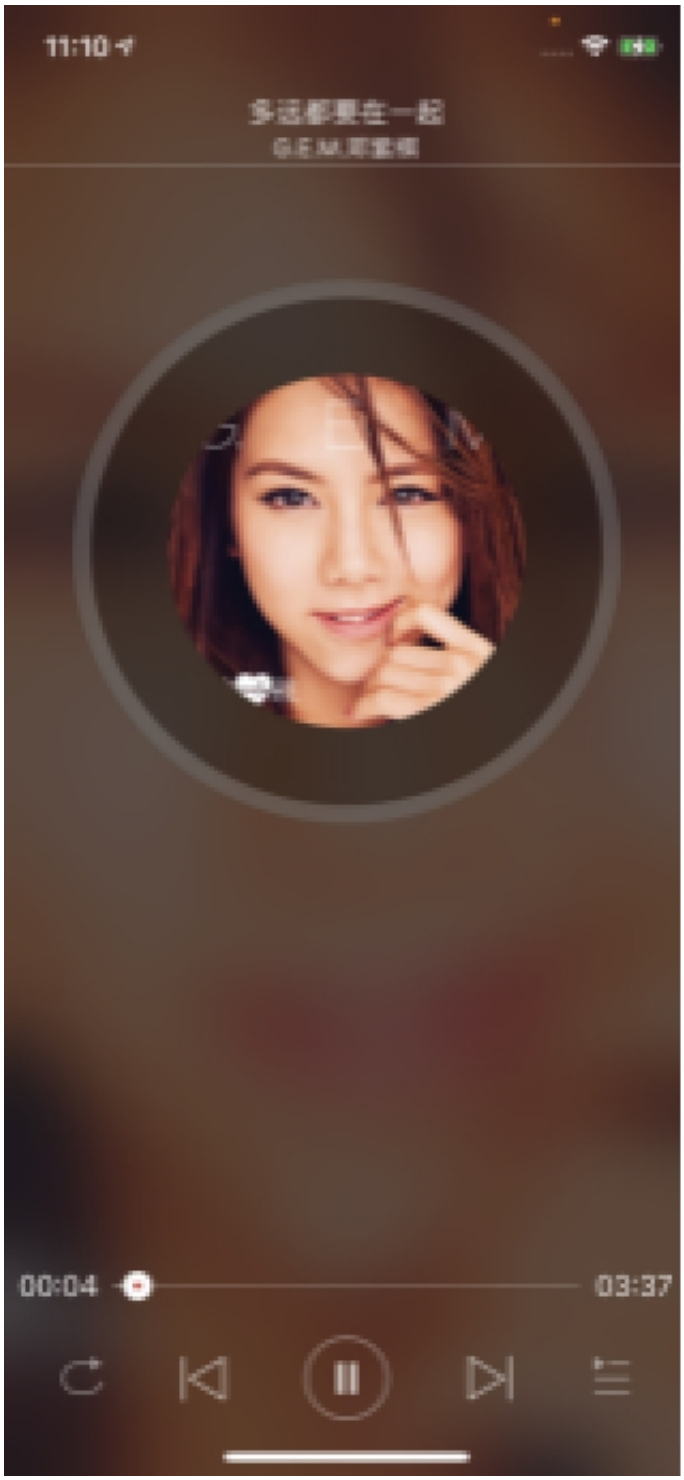
用户也可以点击球体上的文字信息，文字信息会弹框出相关功能的详细介绍。

用户可以旋转球体来展示更多的功能标签。

右上角的蓝色标签信息来提示用户佩戴耳机，效果更佳，用户也可以点击小X号来关闭标签。

左滑界面，为个人中心界面。在这个界面内部可以修改用户的个人信息。下方有两个按钮，分别为我的设备、我的钱包。其中我的设备按钮中可以配置管理SoundAI智能音箱设备。我的钱包中可以将参与答题的余额体现至相关的支付宝账户。

用户可以点击播放列表中的歌曲进行播放，之后会跳转到播放详情界面



此图为播放详情界面，用户可以在此界面查看歌词等信息。还可以切换上下首，以及相关列表的展示和播放模式的设置。用户还可以用语音来控制app的相关操作：“上一首”、“下一首”、“循环播放”、“回到首页”等等一系列的操作。