

# 游戏多媒体引擎

## 高级功能

## 产品文档



腾讯云

**【 版权声明 】**

©2013–2021 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

**【 商标声明 】**

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

**【 服务声明 】**

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

**【 联系我们 】**

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100。

## 文档目录

### 高级功能

范围语音

3D 音效

自定义音频转发路由

实时语音伴奏

实时语音音效

# 高级功能

## 范围语音

最近更新时间：2021-11-03 11:17:58

为方便开发者调试和接入腾讯云游戏多媒体引擎产品 API，这里向您介绍 GME 范围语音的接入技术文档。

GME 范围语音是专门为吃鸡类游戏开发的定制化产品。其有别于普通小队语音房间的核心能力为：

1. 提供吃鸡类游戏、大逃杀类型游戏中特有的“仅小队”或“所有人”的语音模式。
2. 依托于范围判断能力，在同一个语音房间内，支持大量用户同时打开麦克风进行语音通话（范围内用户都可以开启麦克风，本端最多接收20路音频）。

## 基本概念

### 范围语音房间

使用范围语音，需要调用 SetRangeAudioTeamID 接口设置小队号 TeamID，再调用 EnterRoom 接口进入语音房间，即当进入语音房间时指定的 TeamID != 0 时，进入范围语音房间模式。

### 语音模式

当进入范围语音房间时，有两种语音模式可供选择：

| 语音模式 | 参数名称                   | 功能                                  |
|------|------------------------|-------------------------------------|
| 所有人  | RANGE_AUDIO_MODE_WORLD | 设置后玩家附近一定范围的人都能听到该玩家讲话，但不影响小队成员互相通话 |
| 仅小队  | RANGE_AUDIO_MODE_TEAM  | 仅队友可以听到                             |

不同的语音模式下，其具体的声音可达情况如下：

- a. 同一小队内，无论距离多远、玩家为何种语音状态都能听见。
- b. 不同小队情况下，双方开启“所有人”的语音状态情况下，在距离内可以互相听见。

#### 说明

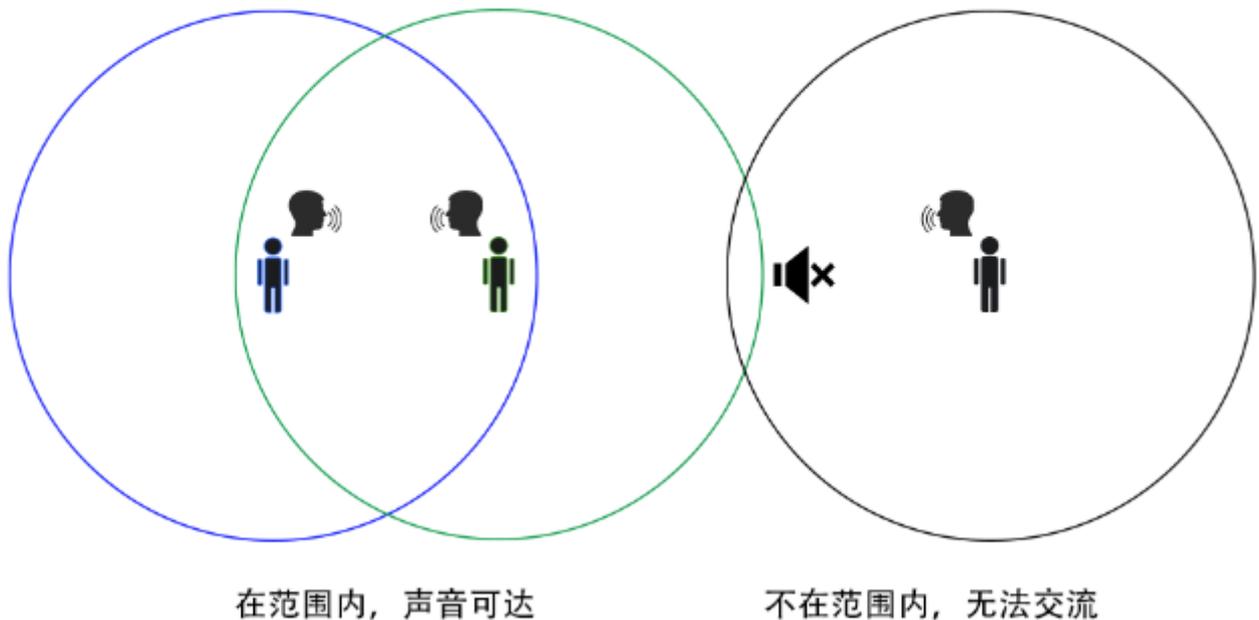
具体玩家声音可达情况请参见 [附录](#)。

### 语音距离范围

如果设置的语音模式为所有人(RANGE\_AUDIO\_MODE\_WORLD)，此时的语音可达范围受 UpdateAudioRecvRange 接口影响。

- 是否在语音距离范围内，不影响同一小队成员互相通话。
- 设置接收语音距离范围参考 API: UpdateAudioRecvRange。
- 支持在范围语音房间中实时切换语音模式。但不支持在房间内更换 TeamID，TeamID 必须在进房前指定。
- 使用范围语音，必须使用流畅音质进房（即 RoomType=1）。

RANGE\_AUDIO\_MODE\_WORLD:



## 使用流程

有别于普通小队语音房间，在使用范围语音能力时，必须使用流畅音质进房，且在调用 EnterRoom 之前，需调用以下两个 API: SetRangeAudioTeamID、SetRangeAudioMode。在进房成功后，调用 UpdateAudioRecvRange，及每帧调用 UpdateSelfPosition。

如果需要同时使用 3D 音效，请在以下步骤1、2 完成后请参见 [同时使用 3D 音效](#)。

### 1. 设置 TeamID

通过此方法设置队伍号，必须在 EnterRoom 之前调用，否则将直接返回错误码：

AV\_ERR\_ROOM\_NOT\_EXITED(1202)。如果是退房后再进房，请在退房成功回调回来之后再调用设置队伍号接口。

#### ⚠ 注意

在退房时，此参数不会自动重置为0，所以一旦决定调用此语音模式，请在每次 EnterRoom 之前都调用此方法设置 TeamID。

## 函数原型

```
ITMGContext SetRangeAudioTeamID(int teamID)
```

| 参数     | 类型  | 意义  |
|--------|-----|---|
| teamID | int | 队伍号，专供范围语音中进行上下行音频流控制。当 TeamID 为0时，通话模式为普通小队语音，默认0。 |

## 2. 设置语音模式

- 通过此方法修改语音模式，可在进房前调用，也可在进房后调用。
- 在进房前调用此方法，方法将影响下一次进房。
- 在进房后调用此方法，将直接改变当前用户的语音模式。
- 在退房时，此参数不会自动重置为 MODE\_WORLD，所以一旦决定调用此方法，请在每次 EnterRoom 之前都调用此方法设置语音模式。

## 函数原型

```
ITMGRoom int SetRangeAudioMode(RANGE_AUDIO_MODE rangeAudioMode)
```

| 参数             | 类型  | 意义   |
|----------------|-----|--|
| rangeAudioMode | int | 0(MODE_WORLD) 代表“所有人”，1(MODE_TEAM) 代表“仅小队” |

## 3. 进入语音房间

在调用 EnterRoom 之前，需调用以下两个 API：SetRangeAudioTeamID、SetRangeAudioMode。

```
ITMGContext.GetInstance(this).EnterRoom(roomId,ITMG_ROOM_TYPE_FLUENCY, authBuffer);
```

一定要使用流畅音质进入语音房间，随后监听进房的回调并进行处理。

```
public void OnEvent(ITMGContext.ITMG_MAIN_EVENT_TYPE type, Intent data) {  
    if (ITMGContext.ITMG_MAIN_EVENT_TYPE.ITMG_MAIN_EVENT_TYPE_ENTER_ROOM == type)  
    {  
        //对事件返回的 Data 进行解析  
        int nErrCode = data.getIntExtra("result", -1);  
    }  
}
```

```
String strErrMsg = data.getStringExtra("error_info");

if (nErrCode == AVErrors.AV_OK)
{
//收到进房信令，进房成功，可以操作设备
ScrollView_ShowLog("EnterRoom success");
Log.i(TAG, "EnterRoom success!");
}
else
{
//进房失败，需分析返回的错误信息
ScrollView_ShowLog("EnterRoom fail :" + strErrMsg);
Log.i(TAG, "EnterRoom fail!");
}
}
}
```

在进房成功后，调用 UpdateAudioRecvRange（至少调用一次），及每帧调用 UpdateSelfPosition。

#### 4. 设置接收语音距离范围

- 通过此方法用于设置接收的语音范围（距离以游戏引擎为准），只支持在**进房成功后调用**。
- 此方法必须配合 UpdateSelfPosition 更新声源方位联合使用。
- 此方法只需调用一次即可生效。

#### 函数原型

```
ITMGRoom int UpdateAudioRecvRange(int range)
```

| 参数    | 类型  | 意义                    |
|-------|-----|-----------------------|
| range | int | 最大可以接收音频的范围，单位为引擎距离单位 |

#### 示例代码

```
ITMGContext.GetInstance().GetRoom().UpdateAudioRecvRange(300);
```

## 5. 更新声源方位

更新声源方位，目的是告诉服务器本端位置，通过本端世界坐标+本端接收音频的范围，与其他端世界坐标+其他端接收音频的范围进行判断，达到范围语音效果。

- 此函数用于更新声源位置信息，只支持在**进房成功后调用**，且需要**每帧调用**，以 Unity 引擎为例，此接口需要在 Update 中调用。
- 使用范围语音一定要更新声源方位，如果不需要范围判断能力，**也需要进房后调用此接口一次**。
- 如果需要同时使用 3D 音效效果，此接口中的参数 axisForward、axisRight 及 axisUp 需要按照下文 [3D 语音部分](#) 进行设置。

### 函数原型

```
public abstract int UpdateSelfPosition(int position[3], float axisForward[3], float axisRight[3], float axisUp[3])
```

| 参数          | 类型      | 意义                    |
|-------------|---------|-----------------------|
| position    | int[]   | 自身在世界坐标系中的坐标，顺序是前、右、上 |
| axisForward | float[] | 在本产品中无需关注             |
| axisRight   | float[] | 在本产品中无需关注             |
| axisUp      | float[] | 在本产品中无需关注             |

## 同时使用 3D 音效

使用范围语音的同时，如果需要同时使用3D音效功能，需要以下几个步骤。

在进房后使用范围语音步骤1、2、3完成后。

### 1. 初始化 3D 音效引擎

此函数用于初始化 3D 音效引擎，在进房后调用。在使用 3D 音效之前必须先调用此接口，只接收 3D 音效而不发出 3D 音效的用户也需要调用此接口。

### 函数原型

```
public abstract int InitSpatializer(string modelPath)
```

| 参数        | 类型     | 意义  |
|-----------|--------|---|
| modelPath | string | 3D 音效资源文件路径，3D 音效模型文件请在此路径 <a href="#">下载</a> ，md5: d0b76aa64c46598788c2f35f5a8a8694，存放于本地中，并通过该参数将存放路径传递给 SDK。必须使用官方提供的文件。 |

## 2. 开启或关闭 3D 音效

此函数用于开启或关闭 3D 音效。开启之后可以听到 3D 音效。

### 函数原型

```
public abstract int EnableSpatializer(bool enable, bool applyToTeam)
```

| 参数          | 类型   | 意义                                 |
|-------------|------|------------------------------------|
| enable      | bool | 开启之后可以听到 3D 音效                     |
| applyToTeam | bool | 3D 语音是否作用于小队内部，仅 enable 为 true 时有效 |

通过 IsEnableSpatializer 接口获取 3D 音效状态。

## 3. 设置接收语音距离范围（3D）

- 通过此方法用于设置接收的语音范围（距离以游戏引擎为准），只支持在**进房成功后调用**。
- 此方法必须配合 UpdateSelfPosition 更新声源方位联合使用。
- 此方法只需调用一次即可生效。
- 3D 音效中，音源音量的大小与音源距离有一定的衰减关系。单位距离超过 range 之后，音量衰减到几乎为零。
- 距离与声音衰减的关系参考文档附录。

### 函数原型

```
ITMGRoom int UpdateAudioRecvRange(int range)
```

| 参数    | 类型  | 意义                    |
|-------|-----|-----------------------|
| range | int | 最大可以接收音频的范围，单位为引擎距离单位 |

### 示例代码

```
ITMGContext.GetInstance().GetRoom().UpdateAudioRecvRange(300);
```

## 4. 更新声源方位 (3D)

更新声源方位，目的是告诉服务器本端位置，通过本端世界坐标+本端接收音频的范围，与其他端世界坐标+其他端接收音频的范围进行判断，达到范围语音效果。

详细技术细节及讲解可参考文章 [《3D 位置语音，引领吃鸡游戏体验升级》](#)。

- 此函数用于更新声源位置信息，只支持在**进房成功后调用**，且需要**每帧调用**，以 Unity 引擎为例，此接口需要在 Update 中调用。
- **请直接复制并调用示例代码使用此功能。**

### 函数原型

```
public abstract int UpdateSelfPosition(int position[3], float axisForward[3], float axisRight[3], float axisUp[3])
```

| 参数          | 类型      | 意义                    |
|-------------|---------|-----------------------|
| position    | int[]   | 自身在世界坐标系中的坐标，顺序是前、右、上 |
| axisForward | float[] | 自身坐标系前轴的单位向量          |
| axisRight   | float[] | 自身坐标系右轴的单位向量          |
| axisUp      | float[] | 自身坐标系上轴的单位向量          |

### 示例代码

#### Unreal

```
FVector cameraLocation = UGameplayStatics::GetPlayerCameraManager(GetWorld(), 0)->GetCameraLocation();
FRotator cameraRotation = UGameplayStatics::GetPlayerCameraManager(GetWorld(), 0)->GetCameraRotation();
int position[] = {
    (int)cameraLocation.X,
    (int)cameraLocation.Y,
    (int)cameraLocation.Z };
```

```
FMatrix matrix = ((FRotationMatrix)cameraRotation);
float forward[] = {
matrix.GetColumn(0).X,
matrix.GetColumn(1).X,
matrix.GetColumn(2).X };
float right[] = {
matrix.GetColumn(0).Y,
matrix.GetColumn(1).Y,
matrix.GetColumn(2).Y };
float up[] = {
matrix.GetColumn(0).Z,
matrix.GetColumn(1).Z,
matrix.GetColumn(2).Z};
ITMGContextGetInstance()->GetRoom()->UpdateSelfPosition(position, forward, right, up);
```

## Unity

```
Transform selftrans = currentPlayer.gameObject.transform;
Matrix4x4 matrix = Matrix4x4.TRS(Vector3.zero, selftrans.rotation, Vector3.one);
int[] position = new int[3] {
selftrans.position.z,
selftrans.position.x,
selftrans.position.y};
float[] axisForward = new float[3] {
matrix.m22,
matrix.m02,
matrix.m12 };
float[] axisRight = new float[3] {
matrix.m20,
matrix.m00,
matrix.m10 };
float[] axisUp = new float[3] {
matrix.m21,
matrix.m01,
matrix.m11 };
ITMGContext.GetInstance().GetRoom().UpdateSelfPosition(position, axisForward, axisRight, axis
Up);
```

## 附录

### 不同语音模式

不同语音模式下，玩家声音可达情况：

- 假设 A 玩家状态为“所有人”，对应 B 玩家在不同语音模式下声音可达情况：

| 是否同一小队 | 是否范围内 | 语音模式       | A与B 是否能相互听到对方的声音 |
|--------|-------|------------|------------------|
| 同一小队   | 是     | MODE_WORLD | 是                |
|        |       | MODE_TEAM  | 是                |
|        | 否     | MODE_WORLD | 是                |
|        |       | MODE_TEAM  | 是                |
| 不同小队   | 是     | MODE_WORLD | 是                |
|        |       | MODE_TEAM  | 否                |
|        | 否     | MODE_WORLD | 否                |
|        |       | MODE_TEAM  | 否                |

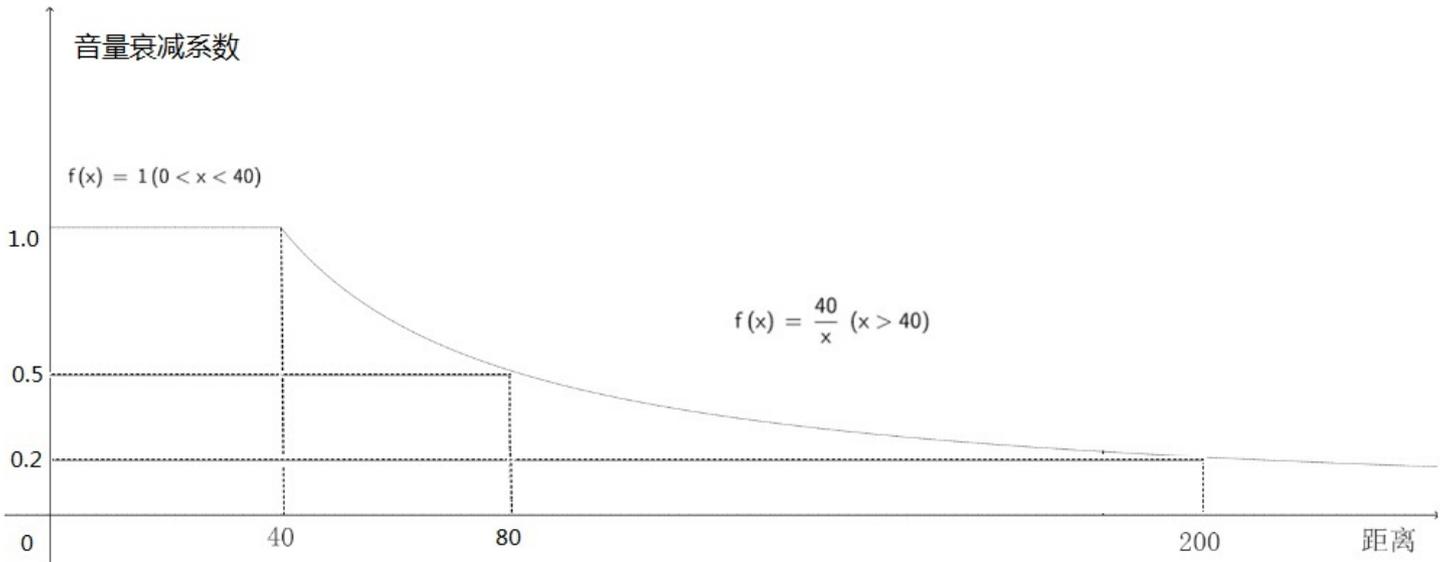
- 假设 A 玩家状态为“仅小队”，对应 B 玩家在不同语音模式下声音可达情况：

| 是否同一小队 | 是否范围内 | 语音状态       | A与B 是否能相互听到对方的声音 |
|--------|-------|------------|------------------|
| 同一小队   | 是     | MODE_WORLD | 是                |
|        |       | MODE_TEAM  | 是                |
|        | 否     | MODE_WORLD | 是                |
|        |       | MODE_TEAM  | 是                |
| 不同小队   | 是     | MODE_WORLD | 否                |
|        |       | MODE_TEAM  | 否                |
|        | 否     | MODE_WORLD | 否                |
|        |       | MODE_TEAM  | 否                |

### 距离与声音衰减的关系

3D 音效中，音源音量的大小与音源距离有一定的衰减关系。单位距离超过 range 之后，音量衰减到几乎为零。

| 距离范围（引擎单位）                | 衰减公式                      |
|---------------------------|---------------------------|
| $0 < N < \text{range}/10$ | 衰减系数：1.0（音量无衰减）           |
| $N \geq \text{range}/10$  | 衰减系数： $\text{range}/10/N$ |



# 3D 音效

最近更新时间：2021-08-12 15:14:48

为方便开发者调试和接入腾讯云游戏多媒体引擎产品 API，这里向您介绍游戏多媒体引擎 3D 音效的接入技术文档。

## 初始化 3D 音效引擎

此函数用于初始化 3D 音效引擎，在进房后调用。在使用 3D 音效之前必须先调用此接口，只接收 3D 音效而不发出 3D 音效的用户也需要调用此接口。

### 函数原型

```
public abstract int InitSpatializer(string modelPath)
```

| 参数        | 类型     | 意义          |
|-----------|--------|-------------|
| modelPath | string | 3D 音效资源文件路径 |

参数中的 3D 音效资源文件，需要另外下载到本地，根据接入 SDK 的版本进行区分：

- 如果是v2.8以下版本，请单击 [下载](#)，md5: d0b76aa64c46598788c2f35f5a8a8694。
- 如果是v2.8及以上版本，请单击 [下载](#)，md5: 3d4d04b3949e267e34ca809e8a0b9243。

SDK 的版本发布历史请参见 [产品动态](#)。

## 开启或关闭 3D 音效

此函数用于开启或关闭 3D 音效。开启之后可以听到 3D 音效。

### 函数原型

```
public abstract int EnableSpatializer(bool enable, bool applyToTeam)
```

| 参数     | 类型   | 意义             |
|--------|------|----------------|
| enable | bool | 开启之后可以听到 3D 音效 |

| 参数          | 类型   | 意义                                |
|-------------|------|-----------------------------------|
| applyToTeam | bool | 3D语音是否作用于小队内部，仅 enable 为 true 时有效 |

## 获取当前 3D 音效状态

此函数用于获取当前 3D 音效状态。

### 函数原型

```
public abstract bool IsEnableSpatializer()
```

| 返回值   | 意义   |
|-------|------|
| true  | 开启状态 |
| false | 关闭状态 |

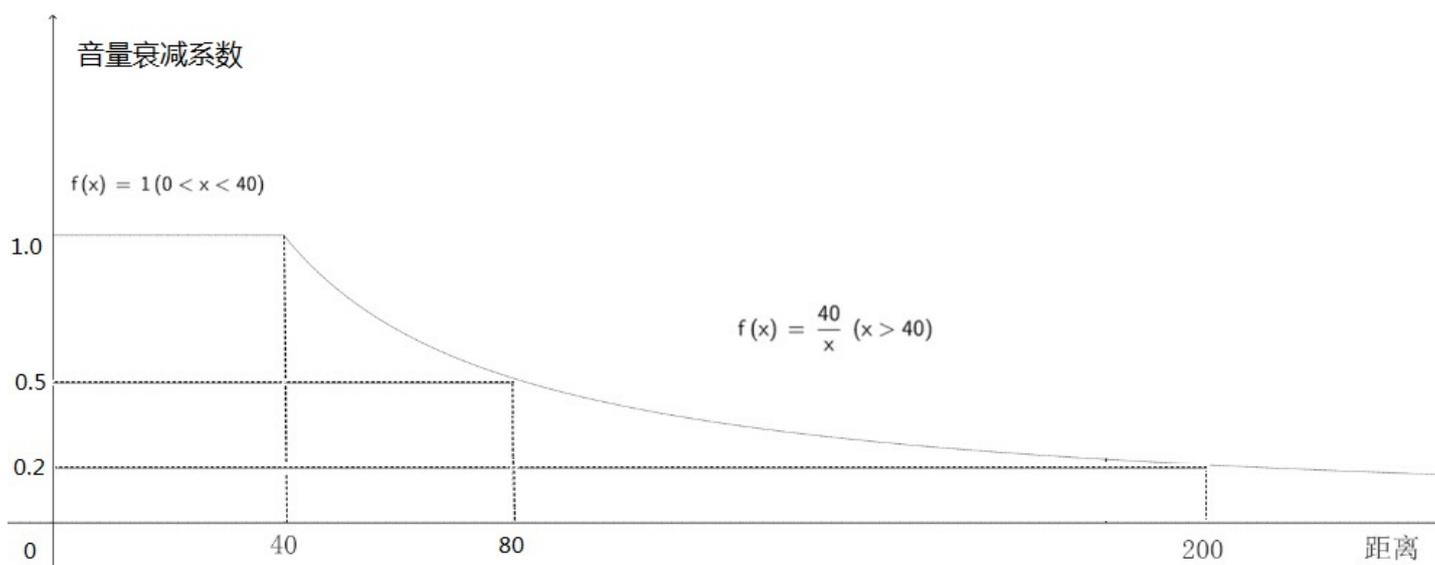
## 更新声源方位（包含朝向）

此函数用于更新声源方位角信息，每帧调用便可实现 3D 音效效果。

### 距离与声音衰减的关系

3D 音效中，音源音量的大小与音源距离有一定的衰减关系。单位距离超过 range 之后，音量衰减到几乎为零。

| 距离范围（引擎单位）                | 衰减公式                      |
|---------------------------|---------------------------|
| $0 < N < \text{range}/10$ | 衰减系数：1.0（音量无衰减）           |
| $N \geq \text{range}/10$  | 衰减系数： $\text{range}/10/N$ |



## 函数原型

```
public abstract void UpdateAudioRecvRange(int range)
```

| 参数    | 类型  | 意义         |
|-------|-----|------------|
| range | int | 设定音效可接收的范围 |

```
public abstract int UpdateSelfPosition(int position[3], float axisForward[3], float axisRight[3], float axisUp[3])
```

| 参数          | 类型      | 意义                    |
|-------------|---------|-----------------------|
| position    | int[]   | 自身在世界坐标系中的坐标，顺序是前、右、上 |
| axisForward | float[] | 自身坐标系前轴的单位向量          |
| axisRight   | float[] | 自身坐标系右轴的单位向量          |
| axisUp      | float[] | 自身坐标系上轴的单位向量          |

## 示例代码

## Unreal

```
FVector cameraLocation = UGameplayStatics::GetPlayerCameraManager(GetWorld(), 0)->GetCameraLocation();
FRotator cameraRotation = UGameplayStatics::GetPlayerCameraManager(GetWorld(), 0)->GetCameraRotation();
int position[] = { (int)cameraLocation.X,(int)cameraLocation.Y, (int)cameraLocation.Z };
FMatrix matrix = ((FRotationMatrix)cameraRotation);
float forward[] = { matrix.GetColumn(0).X,matrix.GetColumn(1).X,matrix.GetColumn(2).X };
float right[] = { matrix.GetColumn(0).Y,matrix.GetColumn(1).Y,matrix.GetColumn(2).Y };
float up[] = { matrix.GetColumn(0).Z,matrix.GetColumn(1).Z,matrix.GetColumn(2).Z};
ITMGContextGetInstance()->GetRoom()->UpdateSelfPosition(position, forward, right, up);
```

## Unity

```
Transform selftrans = currentPlayer.gameObject.transform;
Matrix4x4 matrix = Matrix4x4.TRS(Vector3.zero, selftrans.rotation, Vector3.one);
int[] position = new int[3] { selftrans.position.z, selftrans.position.x, selftrans.position.y };
float[] axisForward = new float[3] { matrix.m22, matrix.m02, matrix.m12 };
float[] axisRight = new float[3] { matrix.m20, matrix.m00, matrix.m10 };
float[] axisUp = new float[3] { matrix.m21, matrix.m01, matrix.m11 };
ITMGContext.GetInstance().GetRoom().UpdateSelfPosition(position, axisForward, axisRight, axisUp);
```

## 相关文档

关于 3D 音效的原理，可以参见 [3D位置语音，引领吃鸡游戏体验升级](#)。

# 自定义音频转发路由

最近更新时间：2021-08-12 15:11:40

为方便 GME 开发者调试和接入腾讯云游戏多媒体引擎产品 API，本文为您介绍适用于 GME 自定义音频转发路由功能的使用参考文档。

## 设置音频转发规则

调用此接口设置音频转发规则，此接口在进房成功回调中调用，调用后此次进房生效，退房后失效。

### 接口原型

```
//iOS接口
-(int)SetServerAudioRouteSendOperateType:(ITMG_SERVER_AUDIO_ROUTE_SEND_TYPE) Sendtype
SendList:(NSArray *)OpenIDForSend RecvOperateType:(ITMG_SERVER_AUDIO_ROUTE_RECV_TYPE) Recvtype
RecvList:(NSArray *)OpenIDForRecv;

//Unity接口
public abstract int SetServerAudioRouteSendOperateType(ITMG_SERVER_AUDIO_ROUTE_SEND_TYPE Sendtype,
string[] OpenIDforSend, ITMG_SERVER_AUDIO_ROUTE_RECV_TYPE Recvtype, string[] OpenIDforRecv);
```

### 类型说明

#### ITMG\_SERVER\_AUDIO\_ROUTE\_SEND\_TYPE

设置发送音频规则，不同的规则填入后，会有不同的发送规则。

| 接收类型                           | 效果   |
|--------------------------------|--|
| AUDIO_ROUTE_NOT_SEND_TO_ANYONE | 本端音频上行发送到后台，但后台不转发给任何人，相当于将自己静音，此时参数 OpenIDForSend 无效，只需填 null |
| AUDIO_ROUTE_SEND_TO_ALL        | 本端音频上行将转发给所有人，此时参数 OpenIDForSend 无效，只需填 null                   |
| AUDIO_ROUTE_SEND_BLACK_LIST    | 本端音频上行将不转发给黑名单的人，黑名单由参数 OpenIDForSend 提供                       |
| AUDIO_ROUTE_SEND_WHITE_LIST    | 本端音频上行将只转发给白名单的人，白名单由参数 OpenIDForSend 提供                       |

 说明:

- 如果类型传入 AUDIO\_ROUTE\_NOT\_SEND\_TO\_ANYONE 以及 AUDIO\_ROUTE\_SEND\_TO\_ALL，此时的参数 OpenIDForSend 不生效，只需要填 null。
- 如果类型传入 AUDIO\_ROUTE\_SEND\_BLACK\_LIST，此时参数 OpenIDForSend 为黑名单列表，最多支持 10 个。
- 如果类型传入 AUDIO\_ROUTE\_SEND\_WHITE\_LIST，此时参数 OpenIDForSend 为白名单列表，最多支持 10 个。

## ITMG\_SERVER\_AUDIO\_ROUTE\_RECV\_TYPE

设置接收音频规则，不同的规则填入后，会有不同的接收规则。

| 接收类型                             | 效果   |
|----------------------------------|--|
| AUDIO_ROUTE_NOT_RECV_FROM_ANYONE | 本端不接受任何音频，相当于关闭房间内扬声器效果，此时参数 OpenIDForSend 无效，只需填 null |
| AUDIO_ROUTE_RECV_FROM_ALL        | 本端接收所有人的音频，此时参数 OpenIDForSend 无效，只需填 null              |
| AUDIO_ROUTE_RECV_BLACK_LIST      | 本端不接收黑名单的人的音频声音，黑名单由参数 OpenIDForSend 提供                |
| AUDIO_ROUTE_RECV_WHITE_LIST      | 本端只接收白名单的人的音频声音，白名单由参数 OpenIDForSend 提供                |

 说明:

- 如果类型传入 AUDIO\_ROUTE\_NOT\_RECV\_FROM\_ANYONE 以及 AUDIO\_ROUTE\_RECV\_FROM\_ALL OpenIDForSend 不生效。
- 如果类型传入 AUDIO\_ROUTE\_RECV\_BLACK\_LIST，此时参数 OpenIDForSend 为黑名单列表，最多支持 10 个。
- 如果类型传入 AUDIO\_ROUTE\_RECV\_WHITE\_LIST，此时参数 OpenIDForSend 为白名单列表，最多支持 10 个。

## 返回值

接口返回值为 QAV\_OK 则表示成功。

- 若回调返回 1004 表示参数错误，建议重新检查参数是否正确。
- 若回调返回 1001 表示重复操作。

- 若回调返回 1201 表示房间不存在，建议检查房间号是否正确。
- 若回调返回 10001 以及 1005 ，建议重新调用接口一次。

更多返回结果解释详情请参见 [错误码](#) 。

## 示例代码

### 执行语句

```
@synthesize _sendListArray;
@synthesize _recvListArray;

int ret = [[[ITMGContext GetInstance] GetRoom] SetServerAudioRouteSendOperateType:SendType SendList:_sendListArray RecvOperateType:RecvType RecvList:_recvListArray];
if (ret != QAV_OK) {
    UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"更新audioroute列表失败" message:[NSString stringWithFormat:@"错误码:%d",ret] delegate:NULL cancelButtonTitle:@"OK" otherButtonTitles:nil];
    [alert show];
}
```

### 回调

```
-(void)OnEvent:(ITMG_MAIN_EVENT_TYPE)eventType data:(NSDictionary *)data{
    NSString* log = [NSString stringWithFormat:@"OnEvent:%d,data:%@", (int)eventType, data];
    switch (eventType) {
        case ITMG_MAIN_EVENT_TYPE_SERVER_AUDIO_ROUTE_EVENT:{
            {
                UIAlertView *alert = [[UIAlertView alloc] initWithTitle:@"更新audioroute" message:[NSString stringWithFormat:@"结果:%@,sub_type: %@ errorinfo: %@", data[@"result"],data[@"sub_type"],data[@"error_info"]] delegate:NULL cancelButtonTitle:@"OK" otherButtonTitles:nil];
                [alert show];
            }
        }
        default:
            break;
    }
}
```

## 获取音频设置转发规则

调用此接口获取音频转发规则。调用后接口返回规则，传入的数组参数会返回相应规则的 openId。

### 接口原型

```

//iOS接口
-(ITMG_SERVER_AUDIO_ROUTE_SEND_TYPE)GetCurrentSendAudioRoute:(NSMutableArray *) OpenIDForSend;
-(ITMG_SERVER_AUDIO_ROUTE_RECV_TYPE)GetCurrentRecvAudioRoute:(NSMutableArray *)OpenIDForRecv;
//Unity接口
public abstract ITMG_SERVER_AUDIO_ROUTE_SEND_TYPE GetCurrentSendAudioRoute(List<string> OpenIDforSend);
public abstract ITMG_SERVER_AUDIO_ROUTE_RECV_TYPE GetCurrentRecvAudioRoute(List<string> OpenIDforRecve);
    
```

### 返回规则

#### ITMG\_SERVER\_AUDIO\_ROUTE\_SEND\_TYPE

| 接收类型                           | 效果                              |
|--------------------------------|---------------------------------|
| AUDIO_ROUTE_NOT_SEND_TO_ANYONE | 本端音频上行发送到后台，但后台不转发给任何人，相当于将自己静音 |
| AUDIO_ROUTE_SEND_TO_ALL        | 本端音频上行将转发给所有人                   |
| AUDIO_ROUTE_SEND_BLACK_LIST    | 本端音频上行将不转发给黑名单的人                |
| AUDIO_ROUTE_SEND_WHITE_LIST    | 本端音频上行将只转发给白名单的人                |
| AUDIO_ROUTE_RECV_INQUIRE_ERROR | 获取出错，检查是否进入房间，是否已经初始化 SDK       |

#### ITMG\_SERVER\_AUDIO\_ROUTE\_RECV\_TYPE

| 接收类型                             | 效果                      |
|----------------------------------|-------------------------|
| AUDIO_ROUTE_NOT_RECV_FROM_ANYONE | 本端不接受任何音频，相当于关闭房间内扬声器效果 |
| AUDIO_ROUTE_RECV_FROM_ALL        | 本端接收所有人的音频              |

| 接收类型                           | 效果                        |
|--------------------------------|---------------------------|
| AUDIO_ROUTE_RECV_BLACK_LIST    | 本端不接收黑名单的人的音频声音           |
| AUDIO_ROUTE_RECV_WHITE_LIST    | 本端只接收白名单的人的音频声音           |
| AUDIO_ROUTE_RECV_INQUIRE_ERROR | 获取出错，检查是否进入房间，是否已经初始化 SDK |

**⚠ 注意：**

在 SetServerAudioRouteSendOperateType 接口中请勿使用 AUDIO\_ROUTE\_RECV\_INQUIRE\_ERROR。

# 实时语音伴奏

最近更新时间：2021-04-27 10:03:31

为方便开发者调试和接入腾讯云游戏多媒体引擎产品 API，本文为您介绍游戏多媒体引擎实时语音伴奏的接入技术文档。

## 实时语音伴奏相关接口

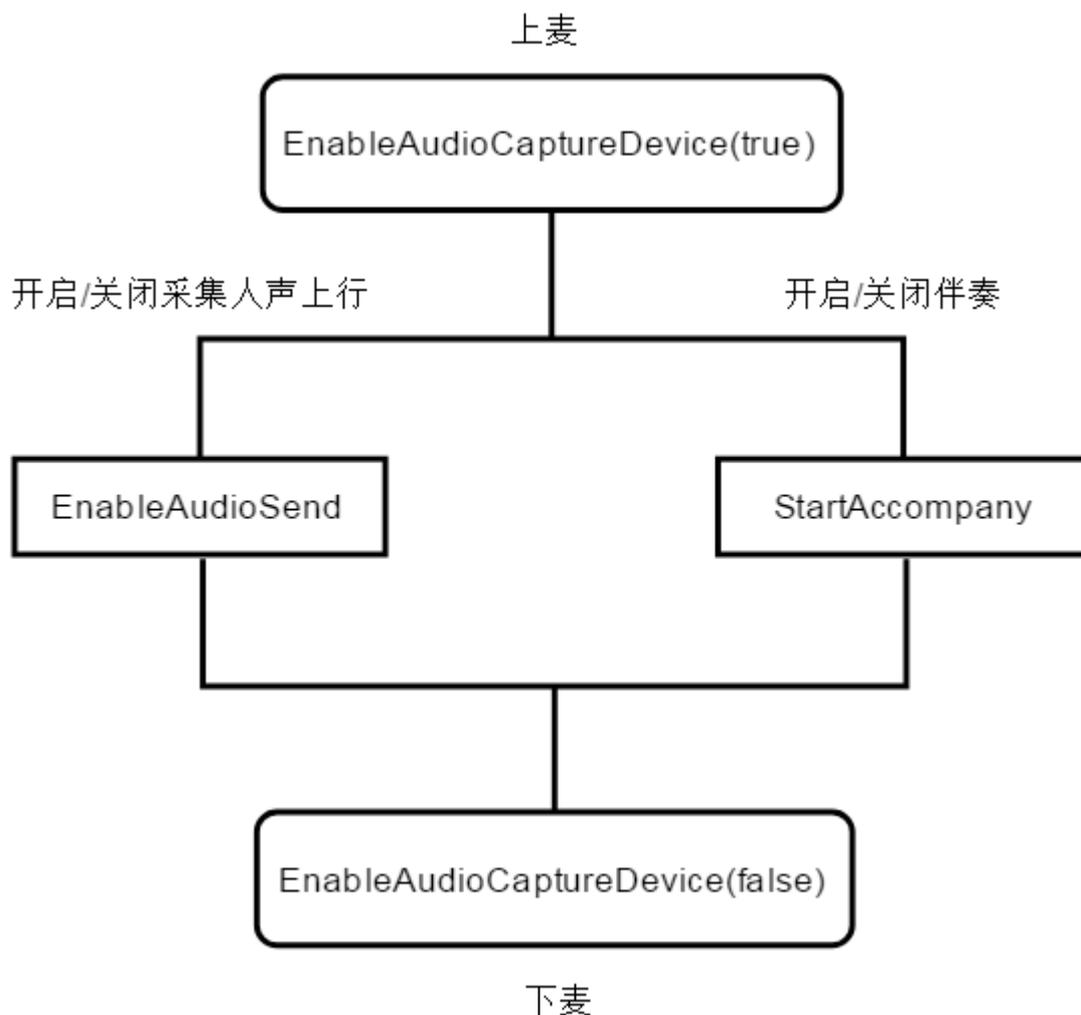
| 接口                                    | 接口含义       |
|---------------------------------------|------------|
| StartAccompany                        | 开始播放伴奏。    |
| StopAccompany                         | 停止播放伴奏。    |
| IsAccompanyPlayEnd                    | 伴奏是否播放完毕。  |
| PauseAccompany                        | 暂停播放伴奏。    |
| ResumeAccompany                       | 重新播放伴奏。    |
| SetAccompanyVolume                    | 设置伴奏音量。    |
| GetAccompanyVolume                    | 获取播放伴奏的音量。 |
| SetAccompanyFileCurrentPlayedTimeByMs | 设置播放进度。    |

### 说明：

如需使用实时语音伴奏，需要在接入 GME SDK 且能在进行实时语音通话的情况下，才可以使用实时语音伴奏。

## 流程图

社交类型 App 调用流程参考图如下：



### 如何配合 EnableAudioCaputreDevice 使用

在进入实时语音房间成功之后，调用 EnableAudioCaputreDevice 打开采集设备，再调用 StartAccompany 播放伴奏。如果需要采集人声，可以调用 EnableAudioSend 实现开麦效果。

### 开始播放伴奏

调用 StartAccompany 接口开始播放伴奏。支持 m4a、wav、mp3 一共三种格式。调用此 API，音量会重置。

### 函数原型

```
ITMGAudioEffectCtrl virtual int StartAccompany(const char* filePath, bool loopBack, int loopCount, int msTime)
```

| 参数 | 类型 | 意义 |
|----|----|----|
|----|----|----|

| 参数        | 类型    | 意义                             |
|-----------|-------|--------------------------------|
| filePath  | char* | 播放伴奏的路径。                       |
| loopBack  | bool  | 是否混音发送，一般都设置为 true，即其他人也能听到伴奏。 |
| loopCount | int   | 循环次数，数值为-1表示无限循环。              |
| msTime    | int   | 延迟时间。                          |

## 示例代码

```
//Windows端代码
ITMGContextGetInstance()->GetAudioEffectCtrl()->StartAccompany(filePath,true,-1,0);
//Android端代码
ITMGContext.GetInstance(this).GetAudioEffectCtrl().StartAccompany(filePath,true,loopCount,0);
//iOS端代码
[[[ITMGContext GetInstance] GetAudioEffectCtrl] StartAccompany:path filePath:isLoopBack loopBack:loopCount msTime:0];
```

## 开始播放伴奏（边下边播）

调用 StartAccompanyDownloading 接口开始边下载边播放伴奏。在代码中实现下载伴奏，未下载完时，可以先将文件路径作为参数传到 StartAccompanyDownloading 里面，可实现边下边播。fileSize 为预估的完整文件大小。调用此接口传入未下载完的文件时，先保证文件至少有10k以上。

## 函数原型

```
ITMGAudioEffectCtrl virtual int StartAccompany(const char* filePath, bool loopBack, int loopCount, int msTime, int fileSize)
```

### 🔗 说明：

iOS 端需要以下配置。

1. 在 iOS 端使用此功能，需要将相关动态库引入工程中，[点击下载 mp3 动态库](#)。
2. 将下载好的文件引入到工程文件中。并在 Link Binary With Libraries 中添加此动态库。
3. 将头文件 TMGEngine\_adv.h 加入工程中，与其他 SDK 头文件同目录下。

## 播放伴奏的回调

开始播放伴奏完成后，回调函数调用 OnEvent，事件消息为 ITMG\_MAIN\_EVENT\_TYPE\_ACCOMPANY\_FINISH，在 OnEvent 函数中对事件消息进行判断。传递的参数 data 包含两个信息，一个是 result，另一个是 file\_path。

### 示例代码

```
void TMGTestScene::OnEvent(ITMG_MAIN_EVENT_TYPE eventType,const char* data){
    switch (eventType) {
    case ITMG_MAIN_EVENT_TYPE_ENTER_ROOM:
    {
        //进行处理
        break;
    }
    ...
    case ITMG_MAIN_EVENT_TYPE_ACCOMPANY_FINISH:
    {
        //进行处理
        break;
    }
    }
}
```

### 停止播放伴奏

调用 StopAccompany 接口停止播放伴奏。

### 函数原型

```
ITMGAudioEffectCtrl virtual int StopAccompany(int duckerTime)
```

| 参数         | 类型  | 意义    |
|------------|-----|-------|
| duckerTime | int | 淡出时间。 |

### 示例代码

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->StopAccompany(0);
```

## 伴奏是否播放完毕

如果播放完毕，返回值为 true，如果没播放完，返回值为 false。

### 函数原型

```
ITMGAudioEffectCtrl virtual bool IsAccompanyPlayEnd()
```

### 示例代码

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->IsAccompanyPlayEnd();
```

## 暂停播放伴奏

调用 PauseAccompany 接口暂停播放伴奏。

### 函数原型

```
ITMGAudioEffectCtrl virtual int PauseAccompany()
```

### 示例代码

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->PauseAccompany();
```

## 恢复播放伴奏

ResumeAccompany 接口用于恢复播放伴奏。

### 函数原型

```
ITMGAudioEffectCtrl virtual int ResumeAccompany()
```

### 示例代码

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->ResumeAccompany();
```

## 设置自己是否可以听到伴奏

此接口用于设置自己是否可以听到伴奏。

### 函数原型

```
ITMGAudioEffectCtrl virtual int EnableAccompanyPlay(bool enable)
```

| 参数     | 类型   | 意义     |
|--------|------|--------|
| enable | bool | 是否能听到。 |

### 示例代码

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->EnableAccompanyPlay(false);
```

## 设置他人是否也可以听到伴奏

设置他人是否也可以听到伴奏。

### 函数原型

```
ITMGAudioEffectCtrl virtual int EnableAccompanyLoopBack(bool enable)
```

| 参数     | 类型   | 意义     |
|--------|------|--------|
| enable | bool | 是否能听到。 |

### 示例代码

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->EnableAccompanyLoopBack(false);
```

## 设置伴奏音量

调用 SetAccompanyVolume 接口设置伴奏音量，默认值为100，数值大于100音量增益，数值小于100音量减益，值域为0 – 200。

### 函数原型

```
ITMGAudioEffectCtrl virtual int SetAccompanyVolume(int vol)
```

| 参数  | 类型  | 意义    |
|-----|-----|-------|
| vol | int | 音量数值。 |

### 示例代码

```
int vol=100;
ITMGContextGetInstance()->GetAudioEffectCtrl()->SetAccompanyVolume(vol);
```

### 获取播放伴奏的音量

GetAccompanyVolume 接口用于获取伴奏音量。

### 函数原型

```
ITMGAudioEffectCtrl virtual int GetAccompanyVolume()
```

### 示例代码

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->GetAccompanyVolume();
```

### 获得伴奏播放进度

以下两个接口用于获得伴奏播放进度。需要注意： $Current / Total =$  当前循环次数， $Current \% Total =$  当前循环播放位置。

### 函数原型

```
ITMGAudioEffectCtrl virtual int GetAccompanyFileTotalTimeByMs()
ITMGAudioEffectCtrl virtual int GetAccompanyFileCurrentPlayedTimeByMs()
```

### 示例代码

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->GetAccompanyFileTotalTimeByMs();
ITMGContextGetInstance()->GetAudioEffectCtrl()->GetAccompanyFileCurrentPlayedTimeByMs();
```

### 设置播放进度

SetAccompanyFileCurrentPlayedTimeByMs 接口用于设置播放进度。

### 函数原型

```
ITMGAudioEffectCtrl virtual int SetAccompanyFileCurrentPlayedTimeByMs(unsigned int time)
```

| 参数   | 类型  | 意义           |
|------|-----|--------------|
| time | int | 播放进度，以毫秒为单位。 |

### 示例代码

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->SetAccompanyFileCurrentPlayedTimeByMs(time);
```

### 设置伴奏音调

SetAccompanyKey 接口用于调整伴奏的音调，在启动伴奏之前调用。

### 函数原型

```
ITMGAudioEffectCtrl virtual int SetAccompanyKey(int nKey)
```

| 参数   | 类型  | 意义                          |
|------|-----|-----------------------------|
| nKey | int | 升降 Key，推荐范围-4到4。当设置为0时为原声调。 |

### 错误码列表

| 错误码名称                              | 错误码值 | 错误码含义    | 解决方法                        |
|------------------------------------|------|----------|-----------------------------|
| QAV_ERR_ACC_OPENFILE_FAILED        | 4001 | 打开文件失败   | 检查文件路径及文件是否存在，检查是否有访问文件的权限。 |
| QAV_ERR_ACC_FILE_FORAMT_NOTSUPPORT | 4002 | 不支持的文件格式 | 检查文件格式是否正确。                 |

| 错误码名称                            | 错误码值 | 错误码含义  | 解决方法                                   |
|----------------------------------|------|--------|--|
| QAV_ERR_ACC_DECODER_FAILED       | 4003 | 解码失败   | 检查文件格式是否正确。                            |
| QAV_ERR_ACC_BAD_PARAM            | 4004 | 参数错误   | 检查代码中所填参数是否正确。                         |
| QAV_ERR_ACC_MEMORY_ALLOC_FAILED  | 4005 | 内存分配失败 | 系统资源耗尽，如果一直存在此错误码，请联系开发人员。             |
| QAV_ERR_ACC_CREATE_THREAD_FAILED | 4006 | 创建线程失败 | 系统资源耗尽，如果一直存在此错误码，请联系开发人员。             |
| QAV_ERR_ACC_STATE_ILLEGAL        | 4007 | 状态非法   | 未处于某种状态，去调用需要处于这个状态才允许调用的接口时，则会产生这个错误。 |

# 实时语音音效

最近更新时间：2021-06-01 10:28:56

为方便开发者调试和接入腾讯云游戏多媒体引擎产品 API，这里向您介绍游戏多媒体引擎实时语音音效的接入技术文档。

## 实时语音音效相关接口

| 接口               | 接口含义      |
|------------------|-----------|
| PlayEffect       | 播放音效      |
| PauseEffect      | 暂停播放音效    |
| PauseAllEffects  | 暂停所有音效    |
| ResumeEffect     | 重新播放音效    |
| ResumeAllEffects | 重新播放所有音效  |
| StopEffect       | 停止播放音效    |
| StopAllEffects   | 停止播放所有音效  |
| SetVoiceType     | 变声特效      |
| SetKaraokeType   | K 歌音效特效   |
| GetEffectsVolume | 获取播放音效的音量 |
| SetEffectsVolume | 设置播放音效的音量 |

### 播放音效

PlayEffect 接口用于播放音效。参数中音效 ID 需要 App 侧进行管理，ID 代表一次独立的播放事件。后续可以根据此 ID 控制此次播放。文件支持 m4a、wav、mp3 一共三种格式。

### 函数原型

```
ITMGAudioEffectCtrl virtual int PlayEffect(int soundId, const char* filePath, bool loop, double pitch, double pan, double gain)
```

| 参数       | 类型     | 意义                               |
|----------|--------|----------------------------------|
| soundId  | int    | 音效 ID                            |
| filePath | char*  | 音效路径                             |
| loop     | bool   | 是否重复播放                           |
| pitch    | double | 播放频率，默认为1.0，该值越小播放速度越慢、时间越长      |
| pan      | double | 声道，取值范围为 -1.0到1.0之间，-1.0表示只开启左声道 |
| gain     | double | 增益音量，取值范围为 0.0到 1.0之间，默认为1.0     |

### 示例代码

```
double pitch = 1.0;
double pan = 0.0;
double gain = 0.0;
//Windows端
ITMGContextGetInstance()->GetAudioEffectCtrl()->PlayEffect(soundId,filePath,true,pitch,pan,gain);
//Android端
ITMGContext.GetInstance(this).GetAudioEffectCtrl().PlayEffect(soundId,filePath,loop);
//iOS端
[[[ITMGContext GetInstance] GetAudioEffectCtrl] PlayEffect:soundId filePath:path loop:isLoop];
```

### 暂停播放音效

PauseEffect 接口用于暂停播放音效。

### 函数原型

```
ITMGAudioEffectCtrl virtual int PauseEffect(int soundId)
```

| 参数      | 类型  | 意义    |
|---------|-----|-------|
| soundId | int | 音效 ID |

### 示例代码

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->PauseEffect(soundId);
```

## 暂停所有音效

调用 `PauseAllEffects` 接口暂停所有音效

### 函数原型

```
ITMGAudioEffectCtrl virtual int PauseAllEffects()
```

### 示例代码

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->PauseAllEffects();
```

## 重新播放音效

`ResumeEffect` 接口用于重新播放音效。

### 函数原型

```
ITMGAudioEffectCtrl virtual int ResumeEffect(int soundId)
```

| 参数      | 类型  | 意义    |
|---------|-----|-------|
| soundId | int | 音效 ID |

### 示例代码

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->ResumeEffect(soundId);
```

## 重新播放所有音效

调用 `ResumeAllEffects` 接口重新播放所有音效。

### 函数原型

```
ITMGAudioEffectCtrl virtual int ResumeAllEffects()
```

## 示例代码

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->ResumeAllEffects();
```

## 停止播放音效

StopEffect 接口用于停止播放音效。

### 函数原型

```
ITMGAudioEffectCtrl virtual int StopEffect(int soundId)
```

| 参数      | 类型  | 意义    |
|---------|-----|-------|
| soundId | int | 音效 ID |

## 示例代码

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->StopEffect(soundId);
```

## 停止播放所有音效

调用 StopAllEffects 接口停止播放所有音效。

### 函数原型

```
ITMGAudioEffectCtrl virtual int StopAllEffects()
```

## 示例代码

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->StopAllEffects();
```

## 变声特效

调用 SetVoiceType 接口设置变声特效。

### 函数原型

```
TMGAudioEffectCtrl int setVoiceType(int type)
```

| 参数   | 类型  | 意义         |
|------|-----|------------|
| type | int | 表示本端音频变声类型 |

| 类型参数                           | 参数代表 | 意义  |
|--------------------------------|------|-----|
| ITMG_VOICE_TYPE_ORIGINAL_SOUND | 0    | 原声  |
| ITMG_VOICE_TYPE_LOLITA         | 1    | 萝莉  |
| ITMG_VOICE_TYPE_UNCLE          | 2    | 大叔  |
| ITMG_VOICE_TYPE_INTANGIBLE     | 3    | 空灵  |
| ITMG_VOICE_TYPE_DEAD_FATBOY    | 4    | 小胖子 |
| ITMG_VOICE_TYPE_HEAVY_MENTA    | 5    | 重金属 |
| ITMG_VOICE_TYPE_DIALECT        | 6    | 歪果仁 |
| ITMG_VOICE_TYPE_INFLUENZA      | 7    | 感冒  |
| ITMG_VOICE_TYPE_CAGED_ANIMAL   | 8    | 困兽  |
| ITMG_VOICE_TYPE_HEAVY_MACHINE  | 9    | 重机器 |
| ITMG_VOICE_TYPE_STRONG_CURRENT | 10   | 强电流 |
| ITMG_VOICE_TYPE_KINDER_GARTEN  | 11   | 幼稚园 |
| ITMG_VOICE_TYPE_HUANG          | 12   | 小顽童 |

### 示例代码

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->setVoiceType(0);
```

### K 歌音效特效

调用 SetKaraokeType 接口设置 K 歌音效特效。

### 函数原型

```
TMGAudioEffectCtrl int SetKaraokeType(int type)
```

| 参数   | 类型  | 意义          |
|------|-----|-------------|
| type | int | 表示本端音频变声类型。 |

| 类型参数                       | 参数代表 | 意义   |
|----------------------------|------|------|
| ITMG_KARAOKE_TYPE_ORIGINAL | 0    | 原声   |
| ITMG_KARAOKE_TYPE_POP      | 1    | 流行   |
| ITMG_KARAOKE_TYPE_ROCK     | 2    | 摇滚   |
| ITMG_KARAOKE_TYPE_RB       | 3    | 嘻哈   |
| ITMG_KARAOKE_TYPE_DANCE    | 4    | 舞曲   |
| ITMG_KARAOKE_TYPE_HEAVEN   | 5    | 空灵   |
| ITMG_KARAOKE_TYPE_TTS      | 6    | 语音合成 |

### 示例代码

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->SetKaraokeType(0);
```

### 获取播放音效的音量

调用 `GetEffectsVolume` 接口获取播放音效的音量，为线性音量，默认值为100，数值大于100为增益效果，数值小于100为减益效果。

### 函数原型

```
ITMGAudioEffectCtrl virtual int GetEffectsVolume()
```

### 示例代码

```
ITMGContextGetInstance()->GetAudioEffectCtrl()->GetEffectsVolume();
```

### 设置播放音效的音量

调用 `SetEffectsVolume` 接口设置播放音效的音量。

### 函数原型

```
ITMGAudioEffectCtrl virtual int SetEffectsVolume(int volume)
```

| 参数     | 类型  | 意义   |
|--------|-----|------|
| volume | int | 音量数值 |

### 示例代码

```
int volume=1;  
ITMGContextGetInstance()->GetAudioEffectCtrl()->SetEffectsVolume(volume);
```

### 相关文档

关于变声玩法的效果请参见 [实时语音趣味变声，大叔变声“妙音娘子”Get一下。](#)