STA-互动助手 SDK 产品白皮书

前言

本产品白皮书为 STA 互动助手 SDK 产品功能的详细介绍与基础开发指南，旨在帮助您更好的了解我们和使用我们的技术服务。STA 互 动助手 SDK 致力于提供包括但不限于切换 Avatar 形象、切换 Avatar 动作、切换相机、切换背景、切换光照、声音驱动口型、声音+文字驱动口型、音频播放等 功能，对于车载系统、社交、直播、教育、金融等领域的多样化应用场景均具备 广泛的适应性，是您互动客服解决方案的理想选择。

一、技术与产品简介

1.技术简介

STA（Speech-to-animation）是虚拟数字人系统的核心图像技术，只需简单输入文字或音频，即可快速生成具备精确 口型、丰富表情和动作的虚拟数字人人播报视频，丰富虚拟客服交互体验。

2.产品简介

移动端互动语音助手是我们自主研发的一套人机智能交互解决方案，该方案为用户提供了一种支持 3D 虚拟形象绘制、语音-口唇动作同步、面部情绪反馈、肢体动作设定的可视化语音助手，极大增强了语音助手产品的生动性和趣味性。

该方案支持文本和语音的输入格式。输入内容会首先经给语音处理单元进行处理，处理完成后将信息返回给语音引擎，由语音引擎输出口唇动作系数，然后再通过 3D 绘制引擎将虚拟形象图像呈现出来。同时可以给形象添加情绪、动作动画设定，完成整个交互操作流程。

1. 产品功能

1. 可支持的模型效果

在 STA 互动助手 SDK 中，支持四大类模型风格：3D 卡通人，3D 卡通动物，2.5D 真人，3D 类真人。四类风格的差异及数据情况如下表所示：

风格类型	示意图片	模型精细度	大小	方案优势	不足
2.5D 真人		6455 四边面	897kb	1、建模成本低、周期短（2~3 周） 2、客户提供素材少（参考 2.5D 规范文档）； 3、真人写实性强，基于照片制作，细节真实； 4、设备性能消耗低，可在低端大屏上运行；	不支持身体动作，头部动作幅度控制在 5% 以内
3D 卡通人		15002 四边面	1.25MB	1、全身建模、支持身体动作； 2、支持风格化定制，包括更换衣服、配饰、发型等； 3、可与 PTA 功能结合，PTA 自生成形象；	卡通风格内，无不足点
3D 卡通动物		3212 四边面	410KB	1、全身建模、支持身体动作； 2、支持包含五官的所有 IP 形象（非人）制作和驱动；	卡通风格内，无不足点
3D 类真人		32472 四边面	13.2MB	1、全身建模，支持全身动作； 2、支持风格化定制，包括更换衣服、配饰、发型等；；	建模周期和成本稍高一点；

可支持调整模型位置大小，支持左右移动、上下移动、并且对模型进行放大缩小。

2. 模型展现力

1. 形象绘制

通过互动语音助手 SDK 可以实现 3D、2.5D 虚拟形象绘制，虚拟形象绘制主要支持口唇动作、表情驱动、肢体动作等功能。在虚拟形象选择方面，用户可以使用已制作的虚拟形象，或者定制相关形象。

当然用户也可以自主制作虚拟形象。我们有标准的形象及动作制作规范，并且提供了易于使用的虚拟形象制作工具，支持用户使用 3D Max 或 Maya 等设计工具设计虚拟形象，最终通过制作工具打包成互动语音助手 SDK 可加载使用的资源文件即可。

2. 口唇动作

口唇动作即虚拟形象说话的口型动作，口型由口型系数驱动。当互动语音助手 SDK 输入为语音时，语音引擎可直接将语音转换为口型系数；当输入为文本时，需要通过 TTS(Text To Speech)，将文本转换为语音，再将语音输入给互动语音助手 SDK，获取口型系数。

将口型系数与虚拟形象模型输入给虚拟形象渲染引擎即可产出生动形象的虚拟形象口唇动作。

2.1.1.3. 表情情绪

表情情绪即虚拟形象的面部表情，其由多个人脸表情单元组合而成。人脸表情单元是采用用于驱动人脸表情的动作单元（共 57 个），通过组合不同的表情单元并调节表情系数，可以涵盖人脸的几乎所有常见面部表情。在口型系数的基础上附加表情可以提供形象的生动性，使形象表情更新丰富。目前已定义好的情绪包括：微笑、悲伤、愤怒，后续会增加更多的表情类别，用户也可以自定义表情。目前虚拟形象支持 42 个中文音素和 39 个英文音素标准口型，经过专业的矫正，达到了口型精准还原。

2.1.1.4. 动作姿态

动作即虚拟形象身体动作，包括头部动作以及肢体动作。动作由骨骼动画实现，不同的形象有不同的骨骼动画，已有的虚拟形象都有对应的动作资源。用户也可以根据动作制作规范制作动作资源。

5. 模型动态物理效果

3D 卡通风格模型在身体驱动时支持自然、模拟现实的头发及服装上的物理效果，大幅度提升模型展现力，模拟现实感。

6. 身体及面部驱动切换自然

模型在动作、情绪之间可以做到自然切换，无卡顿感，无拼接效果。动作及情绪支持自然过渡。

7. 智能标签化控制

针对特定的文本插入配置对应的情绪或者动作标签，后续音频驱动时音频播放到插入标签关键字时可精准驱动该标签匹配的动作和表情。

2. 支持多类型背景切换

支持背景 bundle 切换背景和用户自定义图片资源两周方式切换背景。且支持透明背景及动态背景。

3. 语音兼容性强

可兼容市面上所有主流的 TTS，并支持真人配音版本，只需要传入音频文件即可驱动形象。接口对接方便，根据时间戳开放程度，一般 2 天-2 周即可集成。

二、系统支持及性能介绍

1. 系统支持

1. iOS 支持

iOS 系统提供 objective-c 接口。

支持的机型、系统版本与开发环境：

机型	iPhone 5s 及以上，iPad Air 及以上，iPod touch 第六代及以上
系统版本	IOS 9 及以上
开发环境	Xcode 7 及以上

1.2 Android 支持

Android 系统提供 Java/Kotlin 接口。

支持的系统版本与开发环境：

系统版本	Android 4.2 及以上
开发环境	Android Studio, Android SDK r17 及以上

移动端建议使用的最低配置（运行帧率在 15 帧及以上）：

CPU	高通 骁龙 625 八核 2.0 GHz，ARMV7 及以上
内存	1G 及以上

大屏设备建议使用的最低配置：

CPU	双 Cortex-A72+四 Cortex-A53 大小核 CPU 结构； 频率最高 1.8GHz
内存	4G 内存+16G 储存； 双通道 DDR3-1866/DDR3L-1866/LPDDR3-1866/LPDDR4； 支持 eMMC 5.1，SDIO3.0

1.3 Windows 支持

Windows 系统提供 C/C++/C#接口。

支持的系统版本与开发环境：

系统版本	Windows7 及以上 32 或 64 位系统
开发环境	Visual Studio 2015 及以上

建议使用的最低配置：

内存	4GB 及以上
硬盘	2GB 以上空闲空间
显卡	GT640 及以上

2. 性能测试

1. iOS 版本性能测试

1. iOS 版本性能测试说明

测试版本：iOS sdk1.4

测试环境：使用 3D 卡通女形象，区分纯音频方式驱动和时间戳方式驱动，分别持续播报 5 分钟左右，限制帧率不超过 30

处理耗时、单帧时间、CPU 和内存测试方法：前 300 帧数据不计，取接下来的 3000 帧数据的平均值。

2. iOS 版本性能测试结果

机型	性能指标	3D 卡通
iphoneX	帧率（FPS）	29.5
	渲染耗时（ms）	8.9
	CPU（%）	33.5
	内存（MB）	214.1

	1s 音频处理耗时（ms）	26.8
	时间戳方式一秒耗时（ms）	1.6
iphone7	帧率（FPS）	29.8
	渲染耗时（ms）	6.6
	CPU（%）	22.8
	内存（MB）	176.8
	1s 音频处理耗时（ms）	18.1
	时间戳方式一秒耗时（ms）	1.9
iphone6sp	帧率（FPS）	29.7
	渲染耗时（ms）	11.0
	CPU（%）	38.4
	内存（MB）	103.4
	1s 音频处理耗时（ms）	36.7
	时间戳方式一秒耗时（ms）	2.9
iphone6	帧率（FPS）	29.7
	渲染耗时（ms）	18.9
	CPU（%）	49.2
	内存（MB）	186.3
	1s 音频处理耗时（ms）	35.6
	时间戳方式一秒耗时（ms）	2.2

2. Android 版本性能测试

1. Android 版本性能测试说明

测试版本：Android sdk1.4

测试环境：使用 3D 卡通女形象，区分纯音频方式驱动和时间戳方式驱动，分别持续播报 5 分钟左右，限制帧率不超过 25

处理耗时、单帧时间、CPU 和内存测试方法：前 300 帧数据不计，取接下

来的 1500 帧数据的平均值。

2.2.2 Android 版本性能测试结果

机型	性能指标	3D 卡通
华为 mate9	帧率（FPS）	25
	渲染耗时（ms）	40.3
	CPU（%）	7.6
	内存（MB）	288.8
	1s 音频处理耗时（ms）	300.7
	时间戳方式一秒耗时（ms	17.6
VIVO X9	帧率（FPS）	14.6
	渲染耗时（ms）	71.2
	CPU（%）	9.9
	内存（MB）	268
	1s 音频处理耗时（ms）	86.3
	时间戳方式一秒耗时（ms	23.2
OPPO R9	帧率（FPS）	10.1
	渲染耗时（ms）	103
	CPU（%）	45.3
	内存（MB）	264
	1s 音频处理耗时（ms）	382.7
	时间戳方式一秒耗时（ms	41.2
xiaomi 5c	帧率（FPS）	21.7
	渲染耗时（ms）	47.3
	CPU（%）	18.6
	内存（MB）	158
	1s 音频处理耗时（ms）	770.29

	时间戳方式一秒耗时（ms	28
VIVO x6s	帧率（FPS）	20.7
	渲染耗时（ms）	49.9
	CPU（%）	16.2
	内存（MB）	150.4
	1s 音频处理耗时（ms）	97.9
	时间戳方式一秒耗时（ms	28.6
三星 S6	帧率（FPS）	25
	渲染耗时（ms）	40.4
	CPU（%）	15.3
	内存（MB）	226.6
	1s 音频处理耗时（ms）	177.9
	时间戳方式一秒耗时（ms	17.7
大屏 rockchipZC-339	帧率（FPS）	24.6
	渲染耗时（ms）	41.6
	CPU（%）	19.4
	内存（MB）	163.1
	1s 音频处理耗时（ms）	210.3
	时间戳方式一秒耗时（ms	35.2

3. Windows 版本性能测试

1. Windows 版本性能测试说明

测试版本：Windows sdk1.3

测试环境：使用 3D 卡通女形象，分辨率为 1920*1080，运行半小时后等待 demo 稳定运行，区分纯音频方式驱动和时间戳方式驱动，分别持续播报 5 分钟左右，限制帧率不超过 40

指标获取说明：

1.FPS：通过 fraps 工具记录帧率，每秒记录一次帧率并输出到文件中

2. 渲染耗时：fraps 工具记录帧率，记录每 1 帧渲染的耗时，并记录到文件，后期处理为每 100 帧的平均渲染耗时

3. CPU：使用系统自带性能监控器，获取 demo 进程的% Processor Time/所有进程总的% Processor Time，得出 demo 进程使用的 CPU 百分比

4. 内存：使用系统自带性能监控区，获取 demo 进程的 Working set - Private，得出进程正在使用中内存（单位为字节）

5. 应答效率：处理 30.9 KB (31,738 字节) 音频+文本，记录处理时间并输出到表格

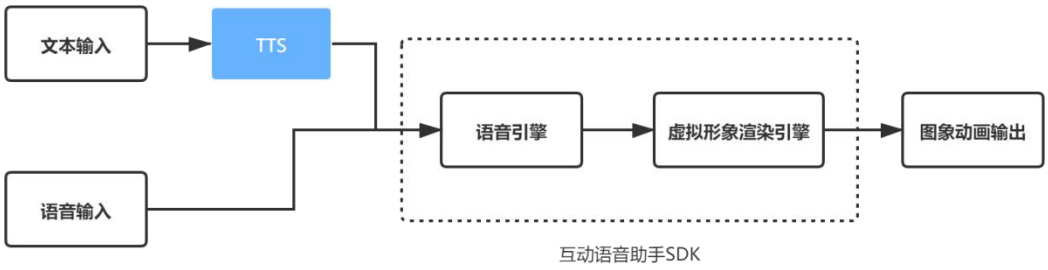
2.3.2 Windows 版本性能测试结果

机型	性能指标	3D 卡通
Win10 64	帧率（FPS）	38.4
	渲染耗时（ms）	26.1
	CPU（%）	6.5
	内存（MB）	592.1
	1s 音频处理耗时（ms）	111.6
	时间戳方式一秒耗时（ms）	0.87
Win10 32	帧率（FPS）	33.7
	渲染耗时（ms）	29.6
	CPU（%）	10
	内存（MB）	184
	1s 音频处理耗时（ms）	131.9
	时间戳方式一秒耗时（ms）	1.15
Win7 64	帧率（FPS）	38.7
	渲染耗时（ms）	25.8
	CPU（%）	8.2
	内存（MB）	100.2

	1s 音频处理耗时（ms）	114.1
	时间戳方式一秒耗时（ms）	0.69
Win7 32	帧率（FPS）	39.3
	渲染耗时（ms）	25.5
	CPU（%）	7.8
	内存（MB）	195.4
	1s 音频处理耗时（ms）	146.6
	时间戳方式一秒耗时（ms）	1.14

三、系统接口定义及说明

1. STA　SDK 技术流程



基本模块如下：

1.　语音引擎

语音引擎是我们自主研发的一套语音转口型的 SDK，集成我们的语音引擎 SDK 可以实现语音、文字、音素到精确口型的转换工作。

2.　虚拟形象渲染引擎

虚拟形象渲染引擎是我们自主研发的 3D 图像绘引擎 SDK，集成该 SDK可以实现虚拟形象绘制，并可驱动虚拟形象实现口唇动作、情绪、动作动画等功

能。

1.3 互动语音助手 SDK

我们通过对语音引擎 SDK、虚拟形象渲染引擎 SDK 进行集成，并对业务逻辑接口进行封装，开了一套易于集成的互动语音助手 SDK。用户在其产品中直接对接该 SDK，目前支持的平台有 Android、iOS、Windows。

2. 接口定义与说明

1. iOS 端接口说明

1. 初始化配置

1. 初始化

```
- (void)setupWithAuthData:(NSData *)authData;
```

接口说明

初始化 FUStaSDK，此方法全局只需要运行一次即可，多次初始化无效。证书需要转换成 NSData 作为参数

参数说明

`authData` 证书，通过读取证书文件内的数组并转换成 NSData 对象

2.1.1.2 加载扩展功能

```
- (void)loadExtensionFunction:(FUStaExtensionFunction )function
                 functionData:(NSData *)functionData
                     authData:(NSData *)authData;
```

接口说明

加载扩展功能，此方法耗时，成功加载一次即可

参数说明

`function` 扩展功能类型，目前有三种功能可选，分别为音频识别、音频校准、中英文解码

`functionData` 扩展功能数据包

`authData` 证书，该功能所需要的证书文件，和初始化证书保持一致

2.1.1.3 运行 SDK

```
- (void)run;
```

接口说明

运行 SDK，SDK 初始化成功后，需要运行此方法才能正常运作

2. 形象渲染

1. 设置渲染对象

```
- (void)setDisplayView:(FUOpenGLView *_Nullable)displayView;
```

接口说明

设置渲染的 OpenGL 视图

参数说明

`displayView` FUOpenGLView 对象，所有道具资源都将渲染在这个视图上

2.1.2.2 设置渲染精度

```
- (void)setRenderQuality:(FUStaRenderQuality)renderQuality;
```

接口说明

设置渲染精度

参数说明

resolutionScale 渲染质量，默认 Middle。与渲染分辨率关系为：Low = 1 x 屏幕分辨率，Middle = 2x，High = 3x

备注

一般情况选择 Middle 即可

2.1.2.3 加载形象

```
- (void)loadAvatar:(FUStaAvatar*)avatar
        isSyncLoad:(BOOL)syncLoad
   completeHandler:(FUAvatarLoadFinishedHandler_nullable)completeHandler
```

接口说明

加载形象

参数说明

`avatar` FUStaAvatar 对象

`syncLoad` 是否同步加载

`completeHandler` 异步加载成功回调

2.1.2.4 设置动画过渡时间

```
- (void)setMotionTransitionDuration:(float)duration;
```

接口说明

设置两个动画过渡时的时间，默认 0.5

参数说明

`duration` 动画过渡时间

2.1.2.5 循环更新动画

```
- (void)updateMotion:(NSString *  nullable)motionPath;
```

接口说明

循环更新动画，执行后，已加载出的形象会重复执行对应动画

参数说明

`motionPath` 动画对应的 bundle 路径

2.1.2.6 定时循环更新动画

```
- (void)updateMotion:(NSString * nullable)motionPath
            duration:(int)duration
     durationHandler:(void (^)(void))durationHandler
```

接口说明

循环更新动画，执行后，已加载出的形象会在指定时间内重复执行对应动画

参数说明

`motionPath` 动画对应的 bundle 路径

`duration` 持续时间

`durationHandler` 回调

2.1.2.7 更新一次动画

```
- (void)updateMotionOnce:(NSString * nullable)motionPath
         completeHandler:(FUMotionUpdateFinishedHandler nullable)completeHandler
```

接口说明

更新一次动画，通过此 API 你可以将多个动画组合在一起

参数说明

`motionPath` 动画对应的 bundle 路径

`completeHandler` 动画执行一次后的回调

2.1.2.8 更新组合道具

```
- (void)updateCombineProp:(FUStaCombinProp *)combinProp
          animationStart:(FUMotionStartPlayHandler nullable)animationStart
          animationFinish:(FUMotionUpdateFinishedHandler nullable)animationFinish;
```

接口说明

更新组合道具，通过此 API 你可以让形象完成某些多道具的动画效果

参数说明

`combinProp` 组合对象

`animationStart` 组合对象中包含的动画开始执行回调，如果有多个道具动画，以形象动画的开始为准

`animationFinish` 组合对象中包含的动画执行完成回调，如果有多个动画，以形象动画的结束为准,无动画时不会执行该回调

备注

此接口目前只支持执行一次组合道具动画

2.1.2.9 移除临时道具

```
- (void)clearTempProp;
```

接口说明

updateCombineProp 可能会引入一些临时道具，通过此接口你可以将其移除

2.1.2.10 暂停当前正在播放的动画

```
- (void)pauseAnimation
```

接口说明

暂停当前正在播放的动画

2.1.2.11 恢复当前正在播放的动画

```
- (void)resumeAnimation
```

接口说明

恢复当前正在播放的动画

2.1.2.12 开启/关闭动画表情

```
- (void)openAnimationExpression:(BOOL)isOpen;
```

接口说明

开启/关闭动画表情，开启后，形象面部表情由动画决定，关闭后，形象面部表情可以通过枚举表情相关接口控制

参数说明

`isOpen` 是否开启

2.1.2.13 瞬时更新静态表情

```
- (void)updateStaticExpression:(FUStaExpression)expression
```

接口说明

瞬时更新静态表情，此方法会在一帧内过度到新的静态表情

参数说明

`expression` 表情枚举值，SDK 内置了九种表情

备注

此接口 1.4.0 版本之后已废弃，你可以使用 updateStaticExpression:occupiedTime:completeHandler 接口实现相同效果，occupiedTime 传 0 即可。

2.1.2.14 自然更新静态表情

```
- (void)updateStaticExpression:(FUStaExpression)expression
                  occupiedTime:(float)occupiedTime
               completeHandler:(FUExpressionUpdateFinishedHandler
__nullable)completeHandler;
```

接口说明

在指定时间内过渡到目标静态表情,此方法仅在 AnimationExpression 关闭时生效

参数说明

`expression` 表情枚举值，SDK 内置了九种表情

`occupiedTime` 表情过度时间

`completeHandler` 表情过渡结束回调

2.1.2.15 瞬时更新动态表情

```
- (void)updateDynamicExpression:(FUStaExpression)expression
```

接口说明

瞬时更新动态表情，此方法会在一帧内过渡到新的动态表情

参数说明

`expression` 表情枚举值，SDK 内置了九种表情

备注

此接口 1.4.0 版本之后已废弃，你可以使用 updateDynamicExpression:occupiedTime:completeHandler 接口实现相同效果，occupiedTime 传 0 即可。

2.1.2.16 自然更新动态表情

```
- (void)updateDynamicExpression:(FUStaExpression)expression
                   occupiedTime:(float)occupiedTime
                completeHandler:(FUExpressionUpdateFinishedHandler
__nullable)completeHandler;
```

接口说明

在指定时间内过渡到目标动态表情,此方法仅在 AnimationExpression 关闭时生效

参数说明

`expression` 表情枚举值，SDK 内置了九种表情

`occupiedTime` 表情过度时间

`completeHandler` 表情过渡结束回调

2.1.2.17 设置说话时表情权重

```
- (void)setWeightToExpressionWhenSpeak:(FUStaExpression)expression
weight:(float)weight;
```

接口说明

设置当形象说话时，同时正在做的表情的权重

参数说明

`expression` 表情枚举值，SDK 内置的九种表情 `weight` 权重，区间 0-1

备注

此接口 1.4.0 版本之后已废弃

2.1.2.18 更新形象位置

```
- (void)updatePosition:(NSArray<NSNumber*>*)position
```

接口说明

更新形象位置，此方法会在一帧内调整形象的位置大小

参数说明

`position` 笛卡尔右手坐标系，需要传入 x、y、z 三点坐标坐标原点是手机屏幕中心，x、y 轴坐标可以调整形象位置，z 轴坐标可以调整形象大小

2.1.2.19 动态更新形象位置

```
- (void)updatePosition:(NSArray<NSNumber*>*)position
translationTime:(float)transLationTime;
```

接口说明

动态更新形象位置，此方法会让形象在指定时间内过渡到指定的位置大小

参数说明

`position` 笛卡尔右手坐标系，需要传入 x、y、z 三点坐标坐标原点是手机屏幕中心，x、y 轴坐标可以调整形象位置，z 轴坐标可以调整形象大小

`transLationTime` 过渡时间

2.1.2.20 左右旋转角色

```
- (void)resetRotDelta:(float)delta;
```

接口说明

左右旋转角色，可以通过此接口配合手势控制角色的旋转

参数说明

`delta` 旋转增量

2.1.2.21 上下平移角色

```
- (void)resetTranslateDelta:(float)delta;
```

接口说明

上下平移角色，可以通过此接口配合手势控制角色的上下平移

参数说明

`delta` 平移增量

2.1.2.22 缩放角色

```
- (void)resetScaleDelta:(float)delta;
```

接口说明

缩放角色，可以通过此接口配合手势控制角色的缩放

参数说明

`delta` 缩放增量

2.1.2.23 通过 Bundle 更新背景

```
- (void)updateBackGroundWithBundlePath:(NSString*)bundlePath;
```

接口说明

通过 Bundle 更新背景

参数说明

`bundlePath` Bundle 路径

2.1.2.24 通过自定义图片更新背景

```
- (void)updateBackGroundWithImage:(UIImage*)image;
```

接口说明

通过自定义图片更新背景

参数说明

`image` UIImage 对象

3. 形象驱动

1. 完整音频配置方法

```
- (void)configureStaAudio:(FUStaAudio*)staAudio
                datasource:(id<FUStaEngineAudioPlayDatasource>
__nullable)audioPlaydatasource
               readyToPlay:(FUStaAudioConfigFinishedHandler_nullable)readyToPlay
                playFinish:(FUStaAudioDriveFinishedHandler_nullable)playFinish
```

接口说明

音频生成口型的配置方法，需要构造 FuStaAudio 对象，FuStaAudio 对象内部会对音频信息进行审核，当音频不符合播放条件时会返回对应错误码此接口耗时，所以已放在异步线程中执行，你可以在 readyToPlay 和 playFinish 中执行你的业务代码

参数说明

`staAudio` FuStaAudio 音频类，可以配置音频相关的信息，详见 FUStaAudio.h

`audioPlaydatasource` 音频播放数据源代理，仅在传 nil 时内置播放器才会生效

`readyToPlay` 口型配置完成时的回调，playAudioWithSdkInlinePlayer 需要再此回调内调用才能正常播放

`playFinish` 口型驱动结束时的回调

2.1.3.2 完整音频开始驱动(内置播放器)

```
- (void)playAudioWithSdkInlinePlayer;
```

接口说明

完整音频的驱动方法，仅在配制方法中的 audioPlayDatasource 为 nil 时有效，此方法将使用 SDK 内置的播放器驱动形象

2.1.3.3 完整音频停止驱动(内置播放器)

```
- (void)stopAudioWithSdkInlinePlayer;
```

接口说明

完整音频的停止方法，仅在配制方法中的 audioPlayDatasource 为 nil 时有效，此方法将使用 SDK 内置的播放器停止

2.1.3.4 完整音频暂停驱动(内置播放器)

```
- (void)pauseAudioWithSdkInlinePlayer;
```

接口说明

完整音频的暂停方法，仅在配制方法中的 audioPlayDatasource 为 nil 时有效，此方法将使用 SDK 内置的播放器暂停

2.1.3.5 完整音频恢复驱动(内置播放器)

```
- (void)resumeAudioWithSdkInlinePlayer;
```

接口说明

完整音频的恢复播放方法，仅在配制方法中的 audioPlayDatasource 为 nil 时有效，此方法将使用 SDK 内置的播放器播放

2.1.3.6 流式驱动

```
- (void)playStaBuffer:(FUStaAudioBuffer * nullable)staAudioBuffer;
```

接口说明

流式音频的驱动方法，需要构造 FUStaAudioBuffer 对象，FUStaAudioBuffer 对象内部会对音频块信息进行检查，当音频块不符合播放条件时会打印对应错误码

参数说明

`staAudioBuffer` pcm 音频块

2.1.3.7 流式驱动结束

```
- (void)playStaBufferCompleted:(FUStaAudioDriveFinishedHandler
__nullable)completeHandler;
```

接口说明

流式音频播放完成接口，此接口可以设置播放完成的回调

参数说明

`completeHandler` 播放结束时的回调

2.1.3.8 重置流式播放器

```
- (void)resetStreamPlayer;
```

接口说明

重置流式播放器，当需要重新播放新的流式音频时，可以通过此方法重置播放器

4. 销毁

1. 停止 SDK

```
- (void)stopRun;
```

接口说明

当不再使用本类时，需要调用此方法释放资源，否则会导致内存泄漏

5. 其他

1. 单例

```
+ (instancetype)shareEngine;
```

接口说明

获取 sdk 单例

2.1.5.2 获取版本信息

```
+ (NSString *)getVersion;
```

接口说明

获取 sdk 版本信息

2.1.5.3 设置日志类型

```
+ (void)setLogLevel:(FULogLevel)logLevel;
```

接口说明

设置日志类型

参数说明

logLevel 分为 FULogLevelNone、FULogLevelDebug、FULogLevelInfo、FULogLevelWarn、FULogLevelError 五个类型，你可以通过逻辑运算符 | 组合你想查看的日志类型

2. Android 端接口说明

1. 配置初始化数据

1. FUSta SDK 配置初始化数据

```
FUStaEngine.Builder builder = new FUStaEngine
        .Builder(context)
        // .setOffLineAuth(byte[]) 加载离线鉴权数据
        .setAuth(authpack.A())
        .setAsrData(byte[])
        .setAlignData(byte[])
        .setFUTtsType(TtsTypeEnum)
        .setCharacterDecoder(bytesDecoder);
mFUStaEngine = builder.build();
```

接口说明

初始化 FUSta，传入上下文（必要），加载网络鉴权字节数组（必要），传入语音识别工具包数据(ASR 方式必须)，传入语音自动校准工具包数据(ALIGN 方式必须)，设置 tts 查询方式(ASR 对应 FUTtsType.ASR，ALIGN 对应 FUTtsType.ALIGNMENT)设置文字编码功能数据文件(ALIGN 方式和文本时间戳需要设置)。

参数说明

`context` 上下文

`setAuth(authpack.A())`字节数组鉴权数据

`setOffLineAuth(byte[])`离线鉴权数据包数据

`setAsrData(byte[])`语音识别工具包数据

`setAlignData(byte[])`语音自动校准工具包数据

`setFUTtsType(TtsTypeEnum type)`tts 查询方式

`setCharacterDecoder(byte[])`文字编码功能数据

备注

App 启动后只需要 SDK 配置初始化数据 一次即可，其中 authpack.A() 鉴权数据声明在 authpack.java 中。必须配置好有效的证书，SDK 才能正常工作。SDK 鉴权两种方式：非离线鉴权与离线鉴权。非离线鉴权只需要设置 authpack.A() 鉴权数据即可。离线鉴权除了设置 authpack.A() 鉴权数据，还需要设置离线鉴权数据包路径：setOffLineAuth(byte[])。根据应用需求，鉴权数据也可以运行时提供（如网络下载），不过要注意证书泄露风险，防止证书被滥用 FUSta SDK 配置初始化数据详见"Android FUStaEngine 集成文档“。

2. 初始化

1. FUSta SDK 初始化

```
void init(STAInitListener listener);
```

接口说明

调用 FUSta SDK 初始化方法，必须放在 FUStaEngine SDK 配置初始化数据之后

参数说明

`FUStaEngine.STAInitListener`SDK 初始化状态的回调，开发者可根据初始化回调进行相关逻辑处理

备注

init 初始化方法，务必在后续操作前调用，否则无法驱动口型。FUSta SDK 初始化详见"Android FUStaEngine 集成文档“。

3. 形象渲染

1. FUSta SDK 创建 GL 环境

```
void onCreate(GLTextureView gLTextureView);
```

2.2.3.2 FUSta SDK 加载形象

```
void selectEffect(FUEffect fuEffect);

void selectEffect(FUEffect fuEffect, FUEffectLoadStateListener
fuEffectLoadStateListener);// 加载形象状态的接口
```

接口说明

加载 FUSta SDK 的操作环境，一般放在 Activity#onCreate 方法中 SDK 加载形象，一定要放在 onCreate 之后

参数说明

`GLTextureView` SDK 内部的 View 组件，负责显示形象

4. 形象驱动

1. FUSta SDK 渲染形象与音频播放（采用 SDK 内部播放器）

```
void startStaDrivingProcess(FUParams params);
```

接口说明

FUSta SDK 渲染形象并播放音频，设置音频自动播放音频并驱动形象对应口型动作绘制。

使用场景：

```
1、流式音频，无时间戳，只支持 pcm
2、流式音频，有时间戳，只支持 pcm
```

```
3、非流式音频，无时间戳，支持 pcm 与 wav
4、非流式音频，有时间戳，支持 pcm 与 wav
```

参数说明

FUParams 为配置类，参数基本设置： 设置 STA 操作为非流式、设置音频数据、设置音频数据类型、设置音频数据对应时间戳等，目前支持 pcm 与 wav

备注

无时间戳，查询口型系数根据音频数据查询，根据机型配置不同耗时也有差别，查询速度跟“有时间戳”查询相比会稍稍耗时。一般查询口型系数方式有 ASR 方式和 ALIGN 方式，ALIGN 方式较快，ASR 方式较慢。有时间戳，查询口型系数根据时间戳查询，速度会更快。

2.2.4.2 SDK 内部播放器控制播放方法

```
void stopMediaPlayer(); // 播放器停止播放，停止驱动口型
```

接口说明

播放器停止，通知 SDK 播放器状态为 `no playing`，停止驱动口型

2.2.4.3 FUSta SDK 渲染形象与音频播放（采用外部播放器）

```
void startStaDrivingProcessNoPlay(FUParams params);
```

接口说明

根据用户传入的外部播放器播放的音频时间戳数据或者 wav/pcm byte[]查询对应的表情系数，与用户自定义播放器播放音频配合使用并且驱动形象对应口型动作等。

参数说明

FUParams 为配置类，参数基本设置： 设置 STA 操作为非流式、设置音频数据、设置音频数据类型、设置音频数据对应时间戳等，目前支持 pcm 与 wav

备注

该方法只是根据用户传入的外部播放器播放的音频时间戳数据或者 wav/pcm byte[]查询对应的表情系数。实现驱动形象对应口型动作还需要下面方法配合使用。

2.2.4.4 FUSta SDK 设置外部播放器播放进度

```
void setOnDrawFrameListener(OnDrawFrameListener onDrawFrameListener);


public interface OnDrawFrameListener {
      /**
     * 获取播放器音频播放进度，开发者在这个方法返回外部播放器的播放进度
     * */
    long onDrawFramePosition();
}
```

接口说明

设置回调接口，onDrawFramePosition()在音频播放状态为 `playing` 时回调，开发者在这个方法返回外部播放器的播放进度。

参数说明

`OnDrawFrameListener` FUSta SDK 每帧绘制时的回调接口。

2.2.4.5 SDK 自定义外部播放器控制播放方法

```
void startMediaPlayer(); // 播放器播放，通知 SDK 播放器状态

void pauseMediaPlayer();  // 播放器暂停，通知 SDK 播放器状态

void stopMediaPlayer(); // 播放器停止播放，通知 SDK 播放器状态
```

接口说明

播放器播放，通知 SDK 播放器状态为 `playing`

播放器暂停，通知 SDK 播放器状态为 `no playing`

播放器停止，通知 SDK 播放器状态为 `no playing`

备注

SDK 状态：`playing` 会渲染音频对应口型，`no playing` 不会渲染口型。

5. 销毁

1. 销毁 FUSta SDK

```
void onDestroy();
```

接口说明

推荐在渲染页面退出时，即在 Activity#onDestroy 方法调用销毁，释放内存资源。

6. 其他

1. 形象切换

```
void selectEffect(FUEffect effect);
```

接口说明

FUSta SDK 的形象道具切换

参数说明

`FUEffect`FUSta 形象

备注

FUEffect 详见 `Android FUStaEngine 集成文档`

2.2.6.2 形象动作切换

```
void updateAnimation(String animPath);// 循环播放
void updateAnimationOnce(String animPath, AnimationStateListener
animationStateListener);// 播放一次
void updateAnimation(FUAction fuAction);// 循环播放
void updateAnimation(FUAction fuAction, AnimationStateListener
animationStateListener);// 播放一次
```

接口说明

FUSta SDK 的形象动作切换

参数说明

`animPath`FUSta 形象动作的路径

`fuAction` 动作组实例

`animationStateListener` 动作播放状态回调接口

2.2.6.3 形象动作暂停播放、恢复播放

```
void resumeAnimation();// 继续播放当前动画
void pauseAnimation();// 暂停播放当前动画
```

接口说明

FUSta SDK 的形象动作播放控制

2.2.6.4 设置动作的过渡时间

```
void setAnimationTransitionDuration(duration);
```

接口说明

FUSta SDK 动作切换有一个过渡，此方法设置动作切换时，原动作和切换动作的过渡时间。

参数说明

`duration` 过渡时间，单位为秒，默认为 0.5s

2.2.6.5 形象表情切换

```
void setEffectMood(FUEmotionType emotionType);//下一帧过渡
void setEffectMood(FUEmotionType emotionType, int fpsNum)//fpsNum 帧过渡
void setEffectMood(FUEmotionType emotionType, int fpsNum, float emotionWeight)//fpsNum
帧过渡并且设置 bs 融合权重(已废弃)
```

接口说明

FUSta SDK 的形象表情切换

参数说明

`emotionType`sdk 支持的表情枚举 FUEmotionType

`fpsNum` 切换表情在多少帧内切换完成，fpsNum >= 0

`emotionWeight` 当前情绪对应 bs 融合权重，区间 0-1

备注

FUEmotionType 支持表情 详见 `Android FUStaEngine 集成文档`

2.2.6.6 展示页面状态同步

```
void onResume();// 进入形象展示页面时调用，非必须
void onPause();// 退出形象展示页面时调用，必须调用
```

接口说明

展示页面状态同步

2.2.6.7 动态切换背景

```
void updateBackground(String bgPath);// 背景切换，参数为 bundle 文件绝对路径或背景图片资
源在 sdcard 的绝对路径
void updateBackground(int resId);// 背景切换，参数为资源 id
```

接口说明

支持背景 bundle 切换背景和用户自定义图片资源 ID 或者在 sdcard 的绝对路径切换背景。

2.2.6.8 调整形象在三维空间中的坐标

```
void setAnimTransX(double x);
void setAnimTransY(double y);
void setAnimTransZ(double z);
void setAnimTransXYZ(double[] xyz);
void setAnimTransXYZ(double[] xyz, int fpsNum);
```

接口说明

FUSta SDK 调整形象在三维空间中的坐标

参数说明

`x` 角色在 X 轴方向的位置，参数:double 一般调整范围 -200~200，效果为左右移动，-200~200 从左到右

`y` 角色在 Y 轴方向的位置，参数:double 一般调整范围 -600~800，效果为上下移动，-600~800 从下到上

`z` 角色在 Z 轴方向的位置，参数:double 一般调整范围 -3000~600，效果为放大缩小，-3000~600 从大到小

`xyz` 角色在三维空间的位置，参数:double 数组 参数表示 x、y、z 轴坐标；fpsNum 表示过渡帧数，效果为统一移动缩放并在 fpsNum 帧内过渡到目标位置和大小

2.2.6.9 旋转形象

```
setRotDelta(double val)
```

接口说明

旋转角色。

参数说明

`val` 表示旋转增量，一般调整范围 -1.0~1.0，效果为旋转

2.2.6.10 缩放形象

```
setScaleDelta(double val)
```

接口说明

缩放角色。

参数说明

`val` 表示缩放增量，一般调整范围 -1.0~1.0，效果为缩放

2.2.6.11 上下移动形象

```
setTranslateDelta(double val)
```

接口说明

上下移动角色。

参数说明

`val` 表示上下增量，一般调整范围 -1.0~1.0，效果为上下移动

2.2.6.12 设置帧率限制

```
setIsLimitFps(boolean)
```

接口说明

FUSta SDK 帧率的限制，初始化 SDK 时设置，默认不限制帧率。

参数说明

true: 限制帧率； false: 不限制帧率

2.2.6.13 设置渲染精度

```
void setResolutionScale(int resolutionScale);
```

接口说明

设置渲染精度挡位

参数说明

resolutionScale 渲染精度挡位，1～3，数值越大，SDK 渲染的图像分辨率越高，但帧率和性能会有所降低，默认 2

备注

一般情况下不建议改动

3. Windwos 端接口说明

1. 创建 OpenGL 环境(重要)

在使用 FaceUnitySTA 对象前，需要创建好 OpenGL 环境。

对于使用 GLFW 框架的 C++版 Demo，可以参考如下示例代码

```
glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_COMPAT_PROFILE);
glfwWindowHint(GLFW_TRANSPARENT_FRAMEBUFFER, 1);
window = glfwCreateWindow(window_width, window_height, "STA example", NULL, NULL);
if (!window) {
    glfwTerminate();
```

```
        exit(EXIT_FAILURE);
    }
    glfwMakeContextCurrent(window);
```

备注：在使用 FUStaEngine SDK 前，需要创建好 OpenGL 环境。

2.3.2 FUStaEngine SDK 初始化数据(重要)

在使用 FUStaEngine SDK 之前，需要对 FUStaEngine SDK 鉴权并初始化。

接口声明

```
        /**
         \brief 初始化并鉴权
         \param sAsetFodrPath 默认数据路径
         \param pAuth 鉴权码
         \param nAuthSz 鉴权码长度
         \param bSetupRndr 是否初始化渲染，true 默认初始化渲染，false 不需要渲染
         \param bOffline 是否进行离线鉴权，默认为否
         \param pOflBdlDt 输入：离线鉴权 bundle 文件数据
                          输出：离线鉴权生成数据
                              默认为空
         \param pOflBdlLen 输入：离线鉴权 bundle 文件数据大小，以字节为单位
                           输出：离线鉴权生成数据大小，以字节为单位
         \return 0 表示失败，非 0 表示成功
         \note 离线鉴权流程：
                 第一次离线鉴权：程序将鉴权码和离线鉴权 bundle 数据发送给服务器，
                                 服务器返回生成的数据（即离线鉴权生成数据）；
                 以后离线鉴权：程序将根据鉴权码和离线鉴权生成数据进行离线鉴权。
        */
       FUSTA_API int32_t Setup(
             const std::string & sAsetFodrPath,
             const char * pAuth, uint32_t nAuthSz,
             bool bSetupRndr = true,
             bool bOffline = false, void ** pOflBdlDt = nullptr, int * pOflBdlLen =
nullptr);
```

备注：初始化和鉴权务必在后续操作前调用，否则无法驱动口型。

2.3.3 FUStaEngine SDK 加载、切换形象(重要)

加载形象、切换要加载的形象的方法。

接口声明

```
        /**
         \brief 更换角色形象
         \param avtrBndlsPath 形象 Bundle 的完整路径
         \exception 该函数可能会抛出异常
         \note 对于大部分角色只要一个形象 bundle，
               但有部分角色（如二次元）需要多个角色 bundle
        */
      FUSTA_API void ChangeAvatar(const std::vector<std::string> & avtrBndlsPath);
```

示例代码

```
        sta.ChangeAvatar({ gsAstDirPath +
"avatar/STA_avatar_kt_def_female_moren.bundle" });
```

4. 形象口型驱动与音频播放(重要)

FUStaEngine SDK 可以通过音频或音频加文本驱动虚拟人口型。使用音频加文本驱动虚拟人口型的准确率更高。

音频分为普通音频和流式音频。普通音频是指一整段音频对应一个音频文件，流式音频是指将一整段音频切分为多个音频文件。

对音频文件要求：

1. 采样率为 16KHz；

2. 单通道；

3. 文件格式为 PCM 或 WAV。

文本文件要求：

1. 内容与音频内容对应；

2. 使用 UTF8 编码；

3. 不能含有特殊符号如℃，否则容易出现口型失真。

2.3.4.1 普通音频播放

接口声明

```
        /**
         \brief 更换音频输入
         \param pWavBufInfo 音频信息
         \param pAudioTxt 音频对应文本，UTF-8 编码，默认值为 nullptr
         \param calbkFunc 音频播放完成的回调函数，默认值为 nullptr
         \exception 该函数可能会抛出异常
        */
       FUSTA_API void ChangeAudio(
             const WavBufferInfo * pWavBufInfo,
             const std::string * pAudioTxt = nullptr,
             std::function<void()> calbkFunc = nullptr);


        /**
         \brief 音频缓存信息
         \note PCM 格式使用结构体定义的 sampleRate,sampleBits,sampleBit 信息
         \note WAV 格式会从数据头中读取
        */
       struct WavBufferInfo {
           WavType wav_type = WavType::WAV;
           const void* wav_buffer = nullptr;
           uint32_t wav_buffer_size = 0;
           uint32_t sample_rate = 16000;  /// 采样频率
           u i n t 3 2 _ t   s a m p l e _ b i t s   =   1 6 ;   / / /   采   样   b i t
           uint32_t sample_channels = 1;  /// 采样通道
           StreamChunkType chunk_type = StreamChunkType::ALL;  /// 流式音频类型，默认
是整段音频
           std::string sFilePath; ///音频文件路径
       };
```

示例代码

```
        std::vector<char> testWAV;
        std::string sAudPath = gsAstDirPath + "audio/tts_part_2.wav";
        loadBinAux(sAudPath, testWAV); // 读取音频文件数据
        std::string testWAVTxt;
        loadTxtAux(gsAstDirPath + "audio/tts_part_2.txt", testWAVTxt);  //读取文本文件数据

        FaceUnity::FaceUnitySTA::WavBufferInfo info; // 创建音频信息对象
        info.wav_type = FaceUnity::FaceUnitySTA::WavType::WAV;
        info.wav_buffer = testWAV.data();
        info.wav_buffer_size = testWAV.size();
        info.sFilePath = sAudPath;
        auto callbackFunc = [=](void) { std::cout << "finish play sound" << std::endl; }; // 创建回调函数
        sta.ChangeAudio(&info, &testWAVTxt, callbackFunc); //播放音频
```

备注

1. 开发者需要保证回调函数中使用的对象或资源在回调函数被调用时仍然是可用的；

2.WavBufferInfo 体中的 chunk_type 是一个枚举类型，用以表示音频的类型。对于普通音频该值应设置为 StreamChunkType::ALL。

2.3.4.2 流式音频播放

接口声明

```
        /**
         \brief 添加语音到待播放队列中
         \param taskID 任务 ID，用于标识播放任务
         \param pWavBufInfo 音频信息
         \param pAudioTxt 音频对应文本，UTF-8 编码，默认值为 nullptr
         \param calbkFunc 音频播放完成的回调函数，默认值为 nullptr
         \exception 该函数可能会抛出异常
        */
        FUSTA_API void AddAudioQueue(const unsigned long taskID,
            const WavBufferInfo* pWavBufInfo,
```

```
                const std::string* pAudioTxt = nullptr,
                std::function<void()> calbkFunc = nullptr);
```

示例代码

```
                unsigned long nPlyTskID = sta.GetPlayAudioTaskID();
                sta.StopSound();
                std::string testWAVTxt;
                loadTxtAux(gsAstDirPath + "audio/audios/whole.txt", testWAVTxt); // 读取整个流式音频对应的文本
                for (int i = 1; i <= 10 &&; ++i)
                {
                     std::vector<char> testWAV;
                     std::string sFilePath(gsAstDirPath);
                     sFilePath.append("audio/audios/").append(std::to_string(i))
                          .append(".wav");
                     loadBinAux(sFilePath, testWAV); // 读取流式音频文件数据
                     FaceUnity::FaceUnitySTA::WavBufferInfo info;  // 创建音频信息对象
                     info.wav_type = FaceUnity::FaceUnitySTA::WavType::WAV;
                     info.wav_buffer = testWAV.data();
                     info.wav_buffer_size = testWAV.size();
                     info.sFilePath = sFilePath;
                     if (i == 1)
                          info.chunk_type = FaceUnity::FaceUnitySTA::StreamChunkType::START;
                     else if (i == 10)
                          info.chunk_type = FaceUnity::FaceUnitySTA::StreamChunkType::END;
                     else
                          info.chunk_type = FaceUnity::FaceUnitySTA::StreamChunkType::NORMAL;
                     auto callbackFunc = [=](void) { std::cout << "finish play " << i << "th sound" << std::endl; }; // 创建回调函数
                     sta.AddAudioQueue(nPlyTskID, &info, &testWAVTxt, callbackFunc); // 播放流式音频
                }
```

备注

1. 开发者需要保证回调函数中使用的对象或资源在回调函数被调用时仍然是可用的；

2. WavBufferInfo 体中的 chunk_type 是一个枚举类型，用以表示音频的类型。对于流式音频，如果是流式音频的开头部分，该值应设置为 StreamChunkType::START；如果是流式音频的中间部分，该值应设置为 StreamChunkType::NORMAL；如果是流式音频的中间部分，该值应设置为 StreamChunkType::END；

3. 老版本中用于播放流式音频的接口 ChangeStreamAudio 现在已经被废弃，请使用 AddAudioQueue 进行替换。

2.3.4.3 暂停播放

接口声明

```
/**
 \brief 暂停播放
 \exception 该函数可能会抛出异常
*/
FUSTA_API void PauseAction();
```

2.3.4.4 恢复播放

接口声明

```
/**
 \brief 继续播放
 \exception 该函数可能会抛出异常
*/
FUSTA_API void ContinueAction();
```

2.3.4.5 停止播放

接口声明

```
/**
```

```
         \brief 停止播放声音
        */
        FUSTA_API void StopSound();
```

2.3.4.6 文本支持 TAG 标签功能

当采用音频加文本的方式播放时，可以在文本中加入特定标记实现在音频的特定位置控制虚拟人执行指定预设动作。

TAG 标签

格式：#%数字#%

实例：#%1#%大家好，#%2#%我是AI数字人。

含义：在播放“大家好”时，虚拟人执行预设动作 1；在播放“我是AI数字人”时，虚拟人执行预设动作 2。

预设动作

功能：定义 TAG 标签中动作与实际动作 bundle 文件的对应关系

存储形式：JSON 文件

格式：

```
{
  "tagMap" : [
    {
      "tag" : "#%1#%",
      "expression" : 0,
      "path" : "animation1_bundle_path"
    },
    {
      "tag" : "#%2#%",
      "expression" : 1,
      "path" : "animation2_bundle_path"
    }
  ]
}
```

说明：

1. tag 字段为使用的标签；

2. expression 字段为虚拟人的表情（0：平静，1：微笑，2：生气，3：悲伤）；

3. path 字段为对应动作 bundle 在资源目录 assets 中的相对路径。

示例：

```
{
  "tagMap" : [
    {
      "tag" : "#%1#%",
      "expression" : 0,
      "path" : "animation/STA_anim_kt_def_female_baoxiong.bundle"
    },
    {
      "tag" : "#%2#%",
      "expression" : 1,
      "path" : "animation/STA_anim_kt_def_female_beishang.bundle"
    }
  ]
}
```

加载预设动作配置：

```
        /**
         \brief 加载 align 特殊标签配置文件
         \param sPath 配置文件路径
         \exception 该函数可能会抛出异常
        */
        FUSTA_API void LoadAlignConfig(const std::string & sPath);
```

备注：播放带有 tag 标签的文本前，需要先加载预设动作配置。

2.3.4.7 设置口型与音频之间的延迟

接口声明

```
    /**
     \brief 设置 TTS 和口型延迟时间
     \param fDltTm 延迟时间，单位秒
    */
FUSTA_API void SetExpDeltaTime(float fDltTm);
```

备注：默认值为 0.06

2.3.4.8 音频时长获取

FUStaEngine SDK 支持获取当前正在播放的音频的时长。

接口声明

```
    /**
     \brief 得到当前音频时长
     \return 总时长，单位为秒
     \exception 该函数可能会抛出异常
    */
FUSTA_API float GetCurrentAudioTotalTime();
```

备注：对于流式音频，该函数返回的值是整个流式音频的时长。

5. 动作切换

切换虚拟人动作有两种方式：直接切换和预加载切换。 预加载切换会先加载后续要执行的动作，后面直接进行播放。 预加载切换相比直接切换会缩短调用 API 到虚拟人开始执行动作之间的时间间隔。

1. 直接切换动作

接口声明

```
    /**
     \brief 更换角色动作
     \param sAnmaBudlPath 动作 Bundle 完整路径
```

```
        \param isLoop 是否循环执行动作，默认值为 true
        \param calbkFunc 执行完动作之后的回调函数，默认值为空
        \exception 该函数可能会抛出异常
       */
      FUSTA_API void ChangeAnimation(const std::string & sAnmaBudlPath,
            bool isLoop = true, std::function<void()> calbkFunc = nullptr);
```

示例代码

```
sta.ChangeAnimation(gsAstDirPath +
"animation/STA_anim_kt_def_female_baoxiong.bundle",
            false, [] {printf("finish animation\n"); fflush(stdout); });
```

备注：开发者需要保证回调函数中使用的对象或资源在回调函数被调用时仍然是可用的。

2.3.5.2 预加载切换动作

接口声明

```
      /**
       \brief 预加载角色动画
       \param sAnimBndlPath 动画 Bundle 完整路径
       \param nEntityID 实体 ID，默认值为 0
       \return 预加载的 Bundle ID
       \exception 该函数可能会抛出异常
       \note 该函数多线程安全
       \note 第一个添加的实体 ID 为 0，第二个添加的实体 ID 为 1，以此类推。
             例如，首先添加人物，再添加道具，则人物的 ID 为 0，道具的 ID 为 1
      */
      FUSTA_API int32_t PreloadAnimation(const std::string & sAnimBndlPath,
                                         const int32_t nEntityID = 0);

      /**
       \brief 播放预加载的动画
       \param nBndlID 动画 Bundle ID
       \param nEntityID 实体 ID，默认值为 0
       \param bLoop 是否循环执行动作，默认值为 true
       \param calbkFunc 执行完动作之后的回调函数，默认值为空
```

```
         \exception 该函数可能会抛出异常
        */
        FUSTA_API void PlayPreloadAnimation(const int32_t nBndlID,
                                            const int32_t nEntityID = 0,
                                            bool bLoop = true,
                                            std::function<void()> calbkFunc =
nullptr);
```

示例代码

```
        std::string sPrldAnimPath = gsAstDirPath
            + "animation/STA_anim_kt_def_female_danshoufangxia.bundle";
        const auto nAniBdlID = sta.PreloadAnimation(sPrldAnimPath, 0);
        sta.PlayPreloadAnimation(nAniBdlID, 0, false, [=]() {
            std::cout << "Play preload animation over, whose path is \""
                << sPrldAnimPath << "\"" << std::endl;
        });
```

2.3.5.3 设置动作的过渡时间

动作切换有一个过渡，此方法设置动作切换时，原动作和切换动作的过渡时间。

接口声明

```
        /*
         \brief 设置动画过渡时间
         \param dTmInSec 动画过渡时间，单位为秒，值不能小于 0
         \exception 该函数可能会抛出异常
        */
        FUSTA_API void SetAnimTransTime(double dTmInSec);
```

2.3.5.4 角色道具动作组合加载

FUStaEngine SDK 支持虚拟人角色、道具和动作的组合加载。

接口声明

```
        /**
         \brief 组合角色动作以及道具和其动作
         \param sAvtrAct 角色动作
         \param sProp 道具
         \param sPropAct 道具动作
         \return 操作是否成功
         \exception 该函数可能会抛出异常
        */
        FUSTA_API void LoadCombineAction(const std::string & sAvtrAct,
             const std::string & sProp, const std::string & sPropAct);
```

示例代码

```
        sta.LoadCombineAction(gsAstDirPath +
"comaction/STA_anim_kt_def_female_changge.bundle",
             gsAstDirPath + "comaction/STA_model_huatong.bundle",
             gsAstDirPath +
"comaction/STA_anim_kt_def_female_model_huatong.bundle");
```

6. 表情切换

表情分为静态表情和动态表情。动态表情相对于静态表情会使虚拟人的表情更为生动。

1. 静态表情切换

接口声明

```
        /* 表情枚举 */
        enum class EExpression
        {
             emNone = 0,  /* 无 情 绪 */
             emSmile = 1, /* 微 笑 */
             emAngry = 2,     /* 生 气 */
             emSad = 3,       /* 悲伤   */
        };
```

```
        /**
         \brief 设置角色表情
         \param exprsn 表情
        */
        FUSTA_API void SetExpression(const EExpression eExprsn);
```

2.3.6.2 动态表情切换

接口声明

```
        /**
         \brief 设置角色动态表情
         \param sPath 动态 bs 文件的路径
         \param bLoop 是否循环播放
         \exception 该函数可能会抛出异常
        */
        FUSTA_API void SetDynExpression(const std::string & sPath,
                                        bool bLoop = false);
```

示例代码

```
sta.SetDynExpression(gsAstDirPath +
"dynamicbs/dynamicbs_cartoon/cartoon_shengqi_bs.txt", true);
```

动态表情文件 目录 assets\dynamicbs 下保存卡通角色（dynamicbs_cartoon）和真人角色（dynamicbs_real）预设的动态表情。

2.3.6.3 表情系数获取

FUStaEngine SDK 支持获取当前虚拟人的表情系数。

接口声明

```
        /**
         \brief 得到当前表情系数
         \param pExprsnIdx 存储当前的表情系数
```

```
     \exception 该函数可能会抛出异常
    */
FUSTA_API void GetCurrentExpressions(float * pExprsnIdx);
```

7. 位置相关

1. 位置切换

FUStaEngine SDK 支持设置虚拟人在空间中的位置。

接口声明

```
/**
 \brief 设置角色在三维空间的位置
 \param dX 位置的 x 坐标，取值范围为[-200, 200]
 \param dY 位置的 y 坐标，取值范围为[-600, 800]
 \param dZ 位置的 z 坐标，取值范围为[-3000, 600]
 \param nEntityID 欲设置的实体 ID，默认值为 0。
 \exception 该函数可能会抛出异常
 \note 第一个添加的实体 ID 为 0，第二个添加的实体 ID 为 1，以此类推。
        例如，首先添加人物，再添加道具，则人物的 ID 为 0，道具的 ID 为 1
*/
FUSTA_API void SetPosition(const double dX,
                           const double dY,
                           const double dZ,
                           int32_t nEntityID = 0);
```

2.3.7.2 鼠标控制设置

FUStaEngine SDK 支持设置指定角色或道具是否受鼠标控制。

接口声明

```
/**
 \brief 设置指定实体是否被鼠标控制
 \param nEntityID 实体 ID
 \param bCtrl 是否被控制
```

```
     \exception 该函数可能会抛出异常
     \note 第一个添加的实体 ID 为 0，第二个添加的实体 ID 为 1，以此类推。
           例如，首先添加人物，再添加道具，则人物的 ID 为 0，道具的 ID 为 1
    */
    FUSTA_API void SetEntityControlByMouse(const int32_t nEntityID
                                           , const bool bCtrl);
```

2.3.7.3 鼠标控制查询

FUStaEngine SDK 支持查询指定角色或道具是否受鼠标控制。

接口声明

```
    /**
     \brief 查询指定实体是否被鼠标控制
     \param nEntityID 实体 ID
     \return 指定实体是否被鼠标控制
     \exception 该函数可能会抛出异常
     \note 第一个添加的实体 ID 为 0，第二个添加的实体 ID 为 1，以此类推。
           例如，首先添加人物，再添加道具，则人物的 ID 为 0，道具的 ID 为 1
    */
    FUSTA_API bool IsEntityControlByMouse(const int32_t nEntityID) const;
```

8. 背景切换

背景切换支持使用 bundle 切换和使用图片切换。

1. 使用 bundle 切换

接口声明

```
    /**
     \brief 更换场景
     \param sScnBndlePath 场景 Bundle 完整路径
     \exception 该函数可能会抛出异常
    */
   FUSTA_API void ChangeScene(const std::string & sScnBndlePath);
```

2.3.8.2 使用图片切换

接口声明

```
/**
 \brief 设置角色背景图片
 \param sPath 图片路径
 \exception 该函数可能会抛出异常
*/
FUSTA_API void SetBkgrndImg(const std::string & sPath);
```

示例代码

```
sta.SetBkgrndImg(gsAstDirPath + "1.jpg");
```

9. 渲染相关

1. 帧率限制

FUStaEngine SDK 支持限制渲染帧率的上限，这样做可以在保证渲染图像流程的情况下减少占用 CPU。

接口声明

```
/**
 \brief 设定渲染帧率
 \param nFPS 帧率，必须是一个正整数
 \exception 该函数可能会抛出异常
*/
FUSTA_API void SetFPS(uint32_t nFPS);
```

备注：

1. 帧率范围为 10 到 60。

2. 此接口只是设置帧率上限，实际帧率与硬件性能、画面尺寸等有关。

2.3.9.2 分辨率设置

FUStaEngine SDK 支持设置渲染分辨率。

接口声明

```
/**
 \brief 设置分辨率
 \param nWidth 分辨率宽，不能为 0
 \param nHeigh 分辨率高，不能为 0
 \note 该操作不改变视角
 \exception 该函数可能会抛出异常
*/
FUSTA_API void SetResolution(uint32_t nWidth, uint32_t nHeigh);
```

2.3.9.3 视口设置

FUStaEngine SDK 支持设置渲染视口。

接口声明

```
/**
 \brief 设置视口大小
 \param nWidth 视口宽，不能为 0
 \param nHeigh 视口高，不能为 0
 \example 例如可以选择 1920*1080 的视角 绘制 3840*2160 的图像
 \exception 该函数可能会抛出异常
*/
FUSTA_API void SetViewPort(uint32_t nWidth, uint32_t nHeigh);
```

2.3.9.4 同时设置分辨率与视口

FUStaEngine SDK 同时支持设置渲染分辨率和视口。

接口声明

```
/**
 \brief 更新窗口大小
     根据窗口大小选择分辨率并选择视角
 \param nWidth 窗口宽
 \param nHeigh 窗口高
 \exception 该函数可能会抛出异常
*/
FUSTA_API void ResizeWindow(uint32_t nWidth, uint32_t nHeigh);
```

2.3.9.5 单帧渲染

FUStaEngine SDK 通过执行 Render()完成一帧画面的渲染工作。

接口声明

```
/**
 \brief 渲染每帧
 \exception 该函数可能会抛出异常
*/
FUSTA_API void Render();
```

2.3.9.6 透明背景

FUStaEngine SDK 支持使用透明背景。

GLFW 框架 在 GLFW 框架下，要实现背景透明，首先需要在创建 GLFW 窗口开启 GLFW_TRANSPARENT_FRAMEBUFFER；其次，需要使用 ChangeScene 加载 assets\scene\aas.bundle。

GLFW 框架示例代码

```
glfwWindowHint(GLFW_CONTEXT_VERSION_MAJOR, 3);
glfwWindowHint(GLFW_CONTEXT_VERSION_MINOR, 3);
glfwWindowHint(GLFW_OPENGL_PROFILE, GLFW_OPENGL_COMPAT_PROFILE);
glfwWindowHint(GLFW_TRANSPARENT_FRAMEBUFFER, 1);
window = glfwCreateWindow(window_width, window_height, "STA Windows Tester 1.8.7",
```

```
NULL, NULL);
    if (!window) {
        glfwTerminate();
        exit(EXIT_FAILURE);
    }
    sta.ChangeScene(gsAstDirPath + "scene/aas.bundle");
```

Qt 框架 在 Qt 框架下，要实现背景透明，首先需要调用 updateFramebufferTransparency 接口；其次，需要使用 ChangeScene 加载 assets\scene\aas.bundle。

接口声明

```
        /**
         \brief 更新窗口帧缓冲区透明度
         \param pWndHdl 窗口句柄
         \return 是否更新成功
         \exception 该函数可能会抛出异常
         \note 该函数用于 Winform
        */
        FUSTA_API void updateFramebufferTransparency(void * pWndHdl);
```

示例代码

```
    try {
        sta.updateFramebufferTransparency(m_pWndHdl);
    } catch (const exception & ex) {
        std::cerr << "Error: Fail to enable transparency, error message is \""
                  << ex.what() << "\"" << std::endl;
    }
    sta.ChangeScene(gsAstDirPath + "scene/aas.bundle");
```

10. 光照相关

1. 光照切换

FUStaEngine SDK 支持切换光照。

接口声明

```
/**
 \brief 更换光照
 \param sLightBndlPath 光照 Bundle 完整路径
 \exception 该函数可能会抛出异常
*/
FUSTA_API void ChangeLight(const std::string & sLightBndlPath);
```

2.3.10.2 阴影开启

FUStaEngine SDK 支持开启虚拟人阴影。

接口声明

```
/**
 \brief 设置是否开启阴影
 \param bEnable 是否开启阴影
 \exception 该函数可能会抛出异常
*/
FUSTA_API void EnableShadow(const bool bEnable);
```

备注：实际的阴影效果与所使用的虚拟人角色有关，有些虚拟人角色开启阴影后效果不明显。

2.3.11 摄像机切换

FUStaEngine SDK 支持切换 OpenGL 中所使用的相机。

接口声明

```
/**
 \brief 更换相机位
 \param sCamBndlPath 相机位 Bundle 完整路径
 \exception 该函数可能会抛出异常
```

```
         */
        FUSTA_API void ChangeCamera(const std::string & sCamBndlPath);
```

12. UI 窗口相关

FUStaEngine SDK 支持显示播放控制相关的 UI 窗口。

1. UI 窗口初始化

接口声明

```
         enum class E_WND_TYPE : int32_t
         {
              eUnknown = 0, ///Unknown window type
              eGlfwWnd = 1, ///GLFW window
              eNatvWnd = 2, ///Native window handle
         };

         /**
          \brief 初始化 Demo UI
          \param pWnd 窗口对象
          \param pGlslVer GLSL 版本，可以为空
          \param eWndType 窗口对象类型
          \exception 该函数可能会抛出异常
         */
        FUSTA_API void InitDemoUI(void * pWnd, const char * pGlslVer,
                                  const E_WND_TYPE eWndType =
E_WND_TYPE::eGlfwWnd);
```

示例代码

```
sta.InitDemoUI(window, "#version 130");
```

备注：接口中窗口对象类型字段用于指定传递给接口的窗口句柄的类型。eGlfwWnd 为 GLFW 窗口句柄，eNatvWnd 为 Window 窗口句柄。

2.3.12.2 UI 窗口开启

接口声明

```
/**
 \brief 控制 Demo UI 开启或关闭
 \param bEnabled 窗口对象
*/
FUSTA_API void ToggleDemoUI(bool bEnabled);
```

示例代码

```
sta.InitDemoUI(window, "#version 130");
sta.ToggleDemoUI(true);
```

备注：如果要开启 UI 窗口，需要先初始化 UI 窗口。

2.3.12.3 UI 窗口使用鼠标查询

FUStaEngine SDK 支持查询 UI 窗口是否在接收鼠标输入。

接口声明

```
/**
 \brief 查询 Demo UI 是否在使用鼠标
 \return Demo UI 是否在使用鼠标
 \note Demo UI 是指角色旁边的对话框
*/
FUSTA_API bool IsDemoUiUseMouse();
```

2.3.12.4 UI 窗口消息处理

FUStaEngine SDK 支持为 UI 窗口设置 Window 消息处理。

接口声明

```
        /**
         \brief 为 Demo UI 设置 Window 消息处理
         \param pWndHdl 窗口句柄
         \param nMsg 消息代码
         \param nWParam windows 消息的 wParam 参数
         \param nLParam windows 消息的 lParam 参数
         \return 消息处理结果
        */
#ifdef _WIN64
        typedef uint64_t FU_WPARAM;
        typedef int64_t FU_LPARAM;
#else
        typedef uint32_t FU_WPARAM;
        typedef int32_t FU_LPARAM;
#endif // _WIN64
        FUSTA_API FU_LPARAM SetWindowProcessHandlerForDemoUi(
            void * pWndHdl, uint32_t nMsg, FU_WPARAM nWParam, FU_LPARAM nLParam);
```

13. 日志相关

1. 日志级别设置

FUStaEngine SDK 支持设置日志级别。

接口声明

```
    /**
     \brief 日志级别
    */
    enum class ELogLevel : int
        {  trace = 0,
        debug = 1,
        info = 2,
        warn = 3,
        error = 4,
        critical = 5,
        off = 6
    };
```

```
/**
 \brief 设置日志级别
 \param eLevel 日志级别
*/
FUSTA_API void SetWinStaLogLevel(const ELogLevel eLevel) noexcept;
```

备注：FUStaEngine SDK 的日志会打印在控制台中。

2.3.13.2 日志级别获取

FUStaEngine SDK 支持获取日志级别。

接口声明

```
/**
 \brief 获取日志级别
 \return 日志级别
*/
FUSTA_API ELogLevel GetWinStaLogLevel() noexcept;
```

14. 其他功能

1. 实体查询

FUStaEngine SDK 支持相关的实体对象是否存在。 实体对象可以是角色对象也可以是道具对象。

接口声明

```
/**
 \brief 查询指定实体是否存在
 \param nEntityID 实体 ID
 \return 指定实体是否存在
 \note 第一个添加的实体 ID 为 0，第二个添加的实体 ID 为 1，以此类推。
        例如，首先添加人物，再添加道具，则人物的 ID 为 0，道具的 ID 为 1
*/
FUSTA_API bool IsEntityExist(const int32_t nEntityID) const;
```

2.3.14.2 版本查询

FUStaEngine SDK 支持查询库版本。

接口声明

```
/**
 \brief 获取版本信息
 \return 版本信息
*/
FUSTA_API const char * GetLibVersion();
```

2.3.14.3 资源销毁

当不再使用 FaceUnitySTA 对象需要销毁其相关的资源。

接口声明

```
  /**
   \brief 销毁资源
  */
 FUSTA_API void Destroy();
```