1. 语音交接文档

Opus是一个完全开放的、免费的、多功能的音频编解码器。 它在交互式的语音和音乐在互联网中的传输方面有着无与伦比的优势，但是同样致力于存储和流媒体应用程序。它是由互联网工程任务组(IETF)制定的标准，标准格式为RFC 6716，由Skype的SILK编解码器和Xiph.Org的CELT编解码器合并发展而来，号称音频编解码器中的瑞士军刀(来自官方视频)。

官方网址：<http://opus-codec.org/>

Opus可以处理广泛的音频应用程序，包括IP电话、视频会议、游戏内聊天、甚至远程现场音乐表演。从低比特率窄带语音到非常高质量的立体声音乐，它都可以适用。技术特点：

* 6 kb /s到510 kb / s的比特率
* 采样率从8 kHz（窄带）到48 kHz（全频）
* 帧大小从2.5毫秒到60毫秒
* 支持恒定比特率（CBR）和可变比特率（VBR）
* 从窄带到全频段的音频带宽
* 支持语音和音乐
* 支持单声道和立体声
* 支持多达255个频道（多数据流的帧）
* 可动态调节比特率，音频带宽和帧大小
* 良好的鲁棒性丢失率和数据包丢失隐藏（PLC）
* 浮点和定点实现

你可以在[RFC 6716](https://tools.ietf.org/html/rfc6716)标准中阅读完整的规范，包括参考实现。也可以在[下载页面](http://opus-codec.org/downloads/" \t "_blank)获得一个最新的Opus 标准。

Libopus是Opus编解码器的参考实现，我们可以参考该代码进行开发。

语音采样最基本的知识了解

<https://www.cnblogs.com/lidabo/p/6846581.html>

* 采样频率：8KHz，可以采集比较完整的语音信息。当然，对于高频的信息无能为力；
* 数据格式：16bit，可以较为详细的表示声音的幅度；
* 声道配置：单声道输入和输出，能够适配所有的机型，少部分手机不支持双声道(立体声)，如果设置立体声会只出现左耳机有声音的情况；
* 音频类型：音乐流
* 输出模式：音频流
* 音频来源：麦克风

int opus\_encoder\_get\_size (int channels)

获取一个OpusEncoder编码器的大小。

- OpusEncoder \* opus\_encoder\_create (opus\_int32 Fs, int channels, int application, int \*error)

分配和初始化一个编码器状态。

- int opus\_encoder\_init (OpusEncoder \*st, opus\_int32 Fs, int channels, int application)

初始化之前分配的编码器指针st指向的内存中的编码器，并且必须通过使用opus\_encoder\_get\_size()方法来返回最小内存大小。

- opus\_int32 opus\_encode (OpusEncoder \*st, const opus\_int16 \*pcm, int frame\_size, unsigned char \*data, opus\_int32 max\_data\_bytes)

编码。

- opus\_int32 opus\_encode\_float (OpusEncoder \*st, const float \*pcm, int frame\_size, unsigned char \*data, opus\_int32 max\_data\_bytes)

编码音频流。

- void opus\_encoder\_destroy (OpusEncoder \*st)

释放一个通过opus\_encoder\_create()分配的OpusEncoder。

- int opus\_encoder\_ctl (OpusEncoder \*st, int request,...)

在Opus编码器中执行CTL函数。

opus\_encoder\_get\_size()返回编码器状态要求的大小。注意,这段代码的未来版本可能改变大小,所以没有assuptions应该对它做出。

　　编码器状态在内存中总是连续,复制它只要一个浅拷贝就足够了。

　　使用opus\_encoder\_ctl()接口可以改变一些编码器的参数设置。所有这些参数都已有缺省值，所以只在必要的情况下改变它们。最常见的参数设置修改是：

opus\_encoder\_ctl(enc, OPUS\_SET\_BITRATE(bitrate));

opus\_encoder\_ctl(enc, OPUS\_SET\_COMPLEXITY(complexity));

opus\_encoder\_ctl(enc, OPUS\_SET\_SIGNAL(signal\_type));

在这里：

　　• bitrate（比特率）的单位是比特/秒(b/s)

　　• complexity（复杂性）是一个值从1到10,1最低，10最高，值越大越复杂

• signal\_type（信号的类型）包括OPUS\_AUTO (缺省), OPUS\_SIGNAL\_VOICE, or OPUS\_SIGNAL\_MUSIC。

Unity3d中如何使用语音以及录制方式

<https://blog.csdn.net/dyc333236081818/article/details/73200608>

Unity3d中如何播放语音

<https://answers.unity.com/questions/702868/how-to-call-a-method-from-onaudiofilterread.html>

实际项目采用了

<https://github.com/BananaHemic/Mumble-Unity>

使用opus进行编码压缩，传输后进行解压缩，最后还原到unity进行播放

可直接下载最新的OPUS源码 android ios的源码自行生成动态库，也可以直接下载网上现有的版本。

Android 的用androidstudio 直接ndk生成即可

Ios 的需要到mac下去生成，以下是生成shell

#!/bin/bash

VERSION="1.1.2" #Opus Version

SDKVERSION="9.2"

MINIOSVERSION="7.0"

# by default, we won't build for debugging purposes

if [ "${DEBUG}" == "true" ]; then

echo "Compiling for debugging ..."

OPT\_CFLAGS="-O0 -fno-inline -g"

OPT\_LDFLAGS=""

OPT\_CONFIG\_ARGS="--enable-assertions --disable-asm"

else

OPT\_CFLAGS="-Ofast -flto -g"

OPT\_LDFLAGS="-flto"

OPT\_CONFIG\_ARGS=""

fi

ARCHS="i386 x86\_64 armv7 armv7s arm64"

DEVELOPER=`xcode-select -print-path`

cd "`dirname \"$0\"`"

REPOROOT=$(pwd)

OUTPUTDIR="${REPOROOT}/out"

mkdir -p ${OUTPUTDIR}/include

mkdir -p ${OUTPUTDIR}/lib

BUILDDIR="${REPOROOT}/build"

echo $REPOROOT

echo $OUTPUTDIR

echo $BUILDDIR

SRCDIR=$REPOROOT

INTERDIR="${BUILDDIR}/built"

mkdir -p $INTERDIR

set +e

CCACHE=`which ccache`

if [ $? == "0" ]; then

echo "Building with ccache: $CCACHE"

CCACHE="${CCACHE} "

else

echo "Building without ccache"

CCACHE=""

fi

set -e

export ORIGINALPATH=$PATH

for ARCH in ${ARCHS}

do

if [ "${ARCH}" == "i386" ] || [ "${ARCH}" == "x86\_64" ]; then

PLATFORM="iPhoneSimulator"

EXTRA\_CFLAGS="-arch ${ARCH}"

EXTRA\_CONFIG=""

else

PLATFORM="iPhoneOS"

EXTRA\_CFLAGS="-arch ${ARCH}"

EXTRA\_CONFIG="--host=arm-apple-darwin"

fi

mkdir -p "${INTERDIR}/${PLATFORM}${SDKVERSION}-${ARCH}.sdk"

./configure --enable-float-approx --disable-shared --enable-static --with-pic --disable-extra-programs --disable-doc ${EXTRA\_CONFIG} \

--prefix="${INTERDIR}/${PLATFORM}${SDKVERSION}-${ARCH}.sdk" \

LDFLAGS="$LDFLAGS ${OPT\_LDFLAGS} -fPIE -miphoneos-version-min=${MINIOSVERSION} -L${OUTPUTDIR}/lib" \

CFLAGS="$CFLAGS ${EXTRA\_CFLAGS} ${OPT\_CFLAGS} -fPIE -miphoneos-version-min=${MINIOSVERSION} -I${OUTPUTDIR}/include -isysroot ${DEVELOPER}/Platforms/${PLATFORM}.platform/Developer/SDKs/${PLATFORM}${SDKVERSION}.sdk" \

# Build the application and install it to the fake SDK intermediary dir

# we have set up. Make sure to clean up afterward because we will re-use

# this source tree to cross-compile other targets.

make -j4

make install

make clean

done

########################################

echo "Build library..."

# These are the libs that comprise libopus.

OUTPUT\_LIBS="libopus.a"

for OUTPUT\_LIB in ${OUTPUT\_LIBS}; do

INPUT\_LIBS=""

for ARCH in ${ARCHS}; do

if [ "${ARCH}" == "i386" ] || [ "${ARCH}" == "x86\_64" ];

then

PLATFORM="iPhoneSimulator"

else

PLATFORM="iPhoneOS"

fi

INPUT\_ARCH\_LIB="${INTERDIR}/${PLATFORM}${SDKVERSION}-${ARCH}.sdk/lib/${OUTPUT\_LIB}"

if [ -e $INPUT\_ARCH\_LIB ]; then

INPUT\_LIBS="${INPUT\_LIBS} ${INPUT\_ARCH\_LIB}"

fi

done

# Combine the three architectures into a universal library.

if [ -n "$INPUT\_LIBS" ]; then

lipo -create $INPUT\_LIBS \

-output "${OUTPUTDIR}/lib/${OUTPUT\_LIB}"

else

echo "$OUTPUT\_LIB does not exist, skipping (are the dependencies installed?)"

fi

done

for ARCH in ${ARCHS}; do

if [ "${ARCH}" == "i386" ] || [ "${ARCH}" == "x86\_64" ];

then

PLATFORM="iPhoneSimulator"

else

PLATFORM="iPhoneOS"

fi

cp -R ${INTERDIR}/${PLATFORM}${SDKVERSION}-${ARCH}.sdk/include/\* ${OUTPUTDIR}/include/

if [ $? == "0" ]; then

# We only need to copy the headers over once. (So break out of forloop

# once we get first success.)

break

fi

done

####################

echo "Building done."

echo "Cleaning up..."

rm -fr ${INTERDIR}

echo "Done."

1. 聊天图文混排

<https://blog.csdn.net/akof1314/article/details/49028279>

源码和示例地址：<https://github.com/akof1314/uGUI_LinkImageText>

1. 语音系统接入文档

ChatManager

OpenMicrophone

短时间录音打开，这们数据会在停止录音时整个包一起发出去

OpenLineMicrophone

长时间录间打开，主要是用来语音通话，数据会在满足包要求即刻发送出去

1. 签到文档

SignInWindow

七是签到主要界面

1. 公告系统

NoticeWindow

公告主要界面

NoticeActivityWindow

公告弹出界面

NoticeSystemWindow

公告系统弹出界面

1. 聊天文档

ChatWindow

聊天主要界面

ChatNormalPanel

左下角聊天界面

ChatMiniPanel

聊天大窗口界面