

《多媒体技术基础》实验

实验指导手册[2024版]

实验要求：以下实验项目选作20学时

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **实验名称** | **建议学时** | **备注** |
| 1 | MCI接口编程 | 4 |  |
| 2 | WAVE文件格式分析 | 4 |  |
| 3 | 基于API的视频卡编程 | 4 |  |
| 4 | JPEG静图像压缩 | 12 |  |
| 5 | Android下的音频编程 | 4 | 2018年起开设 |
| 6 | Android下的摄像头编程 | 4 | 2018年起开设 |
| 7 | Android下使用Tensorflow模型 | 4 | 2018年起开设 |
| 8 | SDL+ffmpeg播放mp4 | 4 | 2020年起开设 |

实验相关示例代码下载

链接：<https://rec.ustc.edu.cn/share/089d4b60-4bfd-11ef-a003-293c134b1311>

密码：eeis@ustc

电子工程与信息科学系

2024年10月30日 16:27

前言

中国科学技术大学是国内最早开设多媒体技术、多媒体通信课程的高校之一。历史上，为了实现对此2门课程教学实验，6系专门建设了多媒体通信实验室。

多媒体通信实验室始建于1993年底，1994年建成。当时的家底是4台386微机，配合声卡、视霸卡及10BASE-2以太网，可以提供基本的多媒体技术和计算机网络技术展示环境。受限于台套数，难以为近百的6系本科生提供教学实验服务。

1998年，多媒体通信实验室接受“211工程”支持进行重建，并由西区电三楼3楼迁至西区电三楼8楼6系“教学实验中心”。在原有设备的基础上，实验室添置100M以太网交换机2台、联想PentiumⅡ300一台、PentiumⅡ266三十台，配置D-Link 100M网卡，安装Windows NT 4.0和Linux（Redhat6.1）操作系统，形成双重网络环境，可适应多种网络实验的需要。所有PC配置了创新16位声卡，2/3的PC还配置了千里眼视频捕捉卡，结合网络环境，可以开发不同复杂度的多媒体应用程序和网络多媒体通信程序，从而形成对6系本科课程《多媒体技术》和研究生课程《多媒体通信》的支持。

2002年，为支持6系研究生课程《信息通信网》，通过编译Linux内核，在国内率先提供了VLAN和IPv6配置的教学环境。由于所支撑的教学实验的变化，实验室更名为多媒体技术与网络通信实验室。

2004年，在“985工程”二期建设经费支持下，实验室进行扩建和升级。增加Cisco路由器3640、3COM三层千兆交换机、Avaya WLAN接入点等网络设备，提供更加灵活的网络环境。同时PC（联想Pentium 4）数量增配至72台，升级多媒体外设（USB接口摄像头、耳麦），此时网卡和声卡已经集成到了PC主板上。

此后，CPU处理能力不断提升、多媒体外设及其接口逐步成熟，PC开始标配多媒体I/O。“旧时王谢堂前燕，飞入寻常百姓家”已成为多媒体技术的真实写照，课程实验的开展不再需要专业的设备、专门的场地，甚至课后自行完成即可。2010年，信息学院成立了信息科学实验教学中心，6系多媒体技术与网络通信实验室及相关教学实验服务逐步并入实验教学中心。

随后，在移动互联网和智能手机技术发展驱动下，基于Android的多媒体开发应用日益普及；同时，2017年最后一届ImageNet挑战赛的落幕，也宣告了深度学习在多媒体信息处理领域的霸主地位。鉴于此，2018年，增设Android下音频、视频、TensorFlow模型调用等3个实验项目。

事实上，伴随着开源软件的盛行，基于操作系统API的多媒体应用程序设计以渐渐难以满足工业界的实际需求。2020年，增设了1个基于开源软件SDL和ffmpeg的实验项目。相信未来仍会有更多开源软件进入课程实验体系。

回顾《多媒体技术基础》课程实验在过去30年的演变历程，恰似一部计算机网络技术与多媒体技术的发展史。由于软件工具的演进与更替，20余年来陆续编写的一些实验示例已难以满足当前教学的需要。2024年暑期，课程组对原有19个实验项目进行“淘汰”和“升级”，精简后保留8个实验项目。作为理论课程的补充，除“JPEG图像压缩”实验项目，其他实验项目均需要课本外扩展内容作为支撑。

陈晓辉

2024年7月25日

目录

[前言 I](#_Toc179417715)

[目录 II](#_Toc179417716)

[1. MCI接口编程 5](#_Toc179417717)

[1.1 实验目的 5](#_Toc179417718)

[1.2 预备知识 5](#_Toc179417719)

[1.3 实验原理 5](#_Toc179417720)

[1.4 实验要求 5](#_Toc179417721)

[1.5 示例代码说明 6](#_Toc179417722)

[1.6 思考题 7](#_Toc179417723)

[2. WAVE文件格式分析 8](#_Toc179417724)

[2.1 实验目的 8](#_Toc179417725)

[2.2 预备知识 8](#_Toc179417726)

[2.3 实验原理 8](#_Toc179417727)

[2.3.1 Windows中的低级多媒体函数 8](#_Toc179417728)

[2.3.2 Wave文件的结构 8](#_Toc179417729)

[2.4 示例代码说明 9](#_Toc179417730)

[2.5 实验要求 11](#_Toc179417731)

[2.6 思考题 11](#_Toc179417732)

[3. 基于WindowsAPI的视频捕捉 12](#_Toc179417733)

[3.1 实验目的 12](#_Toc179417734)

[3.2 预备知识 12](#_Toc179417735)

[3.3 实验原理 12](#_Toc179417736)

[3.3.1 视频捕捉卡/摄像头编程 12](#_Toc179417737)

[3.3.2 视频捕获有关的结构体和API 12](#_Toc179417738)

[3.4 实验要求 14](#_Toc179417739)

[3.5 示例代码说明 14](#_Toc179417740)

[3.6 思考题 16](#_Toc179417741)

[4. JPEG静图像压缩 17](#_Toc179417742)

[4.1 实验目的 17](#_Toc179417743)

[4.2 实验原理 17](#_Toc179417744)

[4.3 实验内容 17](#_Toc179417745)

[4.3.1 DCT 变换 17](#_Toc179417746)

[4.3.2 量化 18](#_Toc179417747)

[4.3.3 熵编码 18](#_Toc179417748)

[4.4 实验要求 19](#_Toc179417749)

[4.5 示例代码 19](#_Toc179417750)

[4.5.1 示例程序的JPEG图像编码过程 19](#_Toc179417751)

[4.5.2 示例程序文件介绍 20](#_Toc179417752)

[4.6 思考题 20](#_Toc179417753)

[5. Android下的音频编程 21](#_Toc179417754)

[5.1 实验目的 21](#_Toc179417755)

[5.2 预备知识 21](#_Toc179417756)

[5.3 实验原理 21](#_Toc179417757)

[5.3.1 Android APIs对多媒体的支持 21](#_Toc179417758)

[5.3.2 WAVE文件格式 22](#_Toc179417759)

[5.3.3 android.media声音播放的几个类 23](#_Toc179417760)

[5.3.4 android.meida录制声音的类 23](#_Toc179417761)

[5.4 示例说明 24](#_Toc179417762)

[5.4.1 示例运行效果 24](#_Toc179417763)

[5.5 实验要求 25](#_Toc179417764)

[5.6 思考题 25](#_Toc179417765)

[5.7 参考文献 26](#_Toc179417766)

[6. Android下的摄像头编程 27](#_Toc179417767)

[6.1 实验目的 27](#_Toc179417768)

[6.2 实验原理 27](#_Toc179417769)

[6.2.1 android.hardware 27](#_Toc179417770)

[6.2.2 android.hardware.Camera类 27](#_Toc179417771)

[6.2.3 Camera2类说明 29](#_Toc179417772)

[6.2.4 android.hardware.ImageReader类 30](#_Toc179417773)

[6.2.5 回调函数 30](#_Toc179417774)

[6.3 示例说明 31](#_Toc179417775)

[6.3.1 示例运行效果 31](#_Toc179417776)

[6.4 实验要求 32](#_Toc179417777)

[6.5 思考题 32](#_Toc179417778)

[6.6 参考文献 32](#_Toc179417779)

[7. Android下使用Tensorflow模型 34](#_Toc179417780)

[7.1 实验目的 34](#_Toc179417781)

[7.2 实验原理 34](#_Toc179417782)

[7.2.1 Python开发环境Anaconda 34](#_Toc179417783)

[7.2.2 Python下使用Tensorflow 34](#_Toc179417784)

[7.2.3 Android JNI 34](#_Toc179417785)

[7.2.4 在Android上集成Tensorflow 35](#_Toc179417786)

[7.2.5 后续Android下对Tensorflow的支持 36](#_Toc179417787)

[7.3 实验原理补充更新2024 36](#_Toc179417788)

[7.4 示例说明 36](#_Toc179417789)

[7.5 实验要求 37](#_Toc179417790)

[7.5.1 Python+Tensorflow环境配置 37](#_Toc179417791)

[7.5.2 Tensorflow例程运行与模型生成 37](#_Toc179417792)

[7.5.3 将Tensorflow下生成的模型集成到Android项目中 37](#_Toc179417793)

[7.6 思考题 38](#_Toc179417794)

[7.7 参考文献 38](#_Toc179417795)

[8. SDL+ffmpeg播放mp4 39](#_Toc179417796)

[8.1 实验目的 39](#_Toc179417797)

[8.2 实验原理 39](#_Toc179417798)

[8.2.1 SDL 39](#_Toc179417799)

[8.2.2 ffmpeg 39](#_Toc179417800)

[8.3 示例说明 39](#_Toc179417801)

[8.4 上机内容 40](#_Toc179417802)

[8.4.1 SDL环境配置 40](#_Toc179417803)

[8.4.2 ffmpeg环境测试 41](#_Toc179417804)

[8.4.3 调试示例代码 42](#_Toc179417805)

[8.4.4 完善示例代码(\*选作) 42](#_Toc179417806)

[8.5 思考题 42](#_Toc179417807)

[8.6 参考文献 42](#_Toc179417808)

[9. 附录：JEPG参数表 43](#_Toc179417809)

[10. 附录：Visual Studio图形界面程序开发 47](#_Toc179417810)

[10.1 概述 47](#_Toc179417811)

[10.1.1 Project & Solution 47](#_Toc179417812)

[10.1.2 MFC 应用程序概述 47](#_Toc179417813)

[10.1.3 Visual Studio安装和配置 48](#_Toc179417814)

[10.2 创建一个MFC应用程序（示例：基于对话框类型） 48](#_Toc179417815)

[10.3 基于对话框的MFC应用程序的UI设计 52](#_Toc179417816)

[10.3.1 常用控件：Static Text，输出提示文本 53](#_Toc179417817)

[10.3.2 常用控件：Edit Control，输入字符串 54](#_Toc179417818)

[10.3.3 常用控件：Button，接受用户动作执行一段代码 54](#_Toc179417819)

[10.4 进阶：了解向导生成的类 55](#_Toc179417820)

[10.5 进阶：添加类变量实现UI控件交互 56](#_Toc179417821)

[10.6 进阶：编译、链接选项的配置 58](#_Toc179417822)

[10.7 VC5/VC6项目移植至高版本Visual Studio 59](#_Toc179417823)

[10.7.1 Visual Studio 2019移植记录 59](#_Toc179417824)

[10.7.2 Visual Studio 2022移植记录 61](#_Toc179417825)

[11. 附录：Visual C++下的多媒体开发 63](#_Toc179417826)

[11.1 Visual C++多媒体开发方法 63](#_Toc179417827)

[11.1.1 使用OLE技术 63](#_Toc179417828)

[11.1.2 运用OCX 63](#_Toc179417829)

[11.1.3 创建和使用DLL 63](#_Toc179417830)

[11.1.4 编制MFC类 63](#_Toc179417831)

[11.1.5 Windows类MCIWnd的应用 63](#_Toc179417832)

[11.1.6 使用Windows多媒体API函数 64](#_Toc179417833)

[11.2 Windows多媒体API函数 64](#_Toc179417834)

[11.2.1 Windows多媒体API函数 64](#_Toc179417835)

[11.2.2 WAVE格式音频的相关函数 64](#_Toc179417836)

[11.2.3 API使用示例 65](#_Toc179417837)

[11.3 Windows中有关多媒体的结构定义 67](#_Toc179417838)

[11.3.1 结构体PCMWAVEFORMAT 67](#_Toc179417839)

[11.3.2 设备头结构WAVEHDR 67](#_Toc179417840)

[12. 附录：Android Studio程序开发极简入门 69](#_Toc179417841)

[12.1 安装Android Studio相关软件 69](#_Toc179417842)

[12.1.1 安装 Android Studio 69](#_Toc179417843)

[12.1.2 安装Android SDK 69](#_Toc179417844)

[12.1.3 关于国内代理 72](#_Toc179417845)

[12.2 创建第一个Android Studio项目 73](#_Toc179417846)

[12.2.1 基础知识：Project与Project的自动构建 73](#_Toc179417847)

[12.2.2 构建一个“Bottom Navigation Views Activity”类型的Project 75](#_Toc179417848)

[12.2.3 进阶：UI界面图元和代码的关联是如何实现的 80](#_Toc179417849)

[12.2.4 进阶：Gradle及build 配置 81](#_Toc179417850)

[12.2.5 进阶：Android Studio各相关软件包的兼容性 85](#_Toc179417851)

[13. 附录：历史上开设过的实验项目 88](#_Toc179417852)

# MCI接口编程

## 实验目的

熟悉Windows下声卡编程的基本过程及原理，掌握Windows下MCI编程的基本知识。

## 预备知识

VisualC++5/6或Visual Studio2019/2022编程方法、MCI编程方法。可参阅“10 附录：Visual Studio图形界面程序开发”和“11 附录：Visual C++下的多媒体开发”。

## 实验原理

设计多媒体程序，关键是对多种媒体设备的控制和使用，在Windows操作系统中，对多媒体设备进行控制主要有三种方法：第一种方法是使用微软公司窗口系统中对多媒体支持的MCI，即媒体控制接口，MCI是多媒体设备和多媒体应用软件之间进行设备无关的沟通的桥梁。在VB和VC中，MCI都得到了很好的支持。第二种方法，通过调用Windows的API(应用程序接口)多媒体相关函数实现媒体控制。第三种方法是使用OLE(Object Linking ＆ Embedding)，即对象链接与嵌入技术，它为不同软件之间共享数据和资源提供了有力的手段。

MCI 为应用程序提供与设备无关的功能，用于控制音频和视频外围设备。 应用程序可以使用 MCI 来控制任何受支持的多媒体设备，包括波形音频设备、MIDI 排序器、CD 音频设备和数字视频 (视频播放) 设备。

需要注意，MCI 是旧版功能。 它已被 MediaPlayer 取代。 MediaPlayer 已针对Windows 10和Windows 11进行了优化。 Microsoft 强烈建议新代码尽可能使用 MediaPlayer 而不是 MCI。（<https://learn.microsoft.com/zh-cn/windows/win32/multimedia/mci-command-strings-and-messages> ）

我们在本实验中用最简单的方法来实现声音的播放和录制—媒体控制接口编程。有关知识请阅读附录中“VC多媒体”部分。【提示】编制自己的多媒体程序时需要包括头文件mmsystem.h，链接winmm.lib库。

## 实验要求

本实验非常简单和基础，但是对VC或Visual Studio图形化程序设计有一定程度的要求，做实验之前请自行学习MFC编程（Visual Studio下MFC程序设计可参考10 附录：Visual Studio图形界面程序开发）。实验具体要求如下：

[1-1] 学会使用PlaySound()、SndPlaySound() 来播放WAV文件。

[1-2] 查阅Microsoft Learn网站，了解mciSendCommand() 和 mciSendString()的差异。

<https://learn.microsoft.com/en-us/windows/win32/multimedia/mci-device-types>

[1-3] 学会使用mciSendCommand() 或mciSendString() 来播放WAV文件。

[1-4] 学会使用mciSendCommand() 或mciSendString() 来录制WAV文件。

[1-5] 查阅Microsoft Learn网站，了解Windows支持的MCI设备类型。

<https://learn.microsoft.com/en-us/windows/win32/multimedia/mci-device-types>

[1-6] 查阅Windows下注册表，了解不同媒体文件的后缀及其设备类型标识字符串（如waveaudio、sequencer等）：

HKEY\_LOCAL\_MACHINE/Software/Microsoft/Windows NT/CurrentVersion/MCI Extensions

[1-7] 查阅Microsoft Learn网站，了解Windows支持的多媒体命令。

<https://learn.microsoft.com/zh-cn/windows/win32/multimedia/multimedia-commands>

[1-8] 制作一个媒体播放器，实现对WAV、MIDI以及MP3声音文件的播放、暂停、继续、停止。

## 示例代码说明

|  |  |
| --- | --- |
| VC6版本的示例 |  |
| Visual Studio 2019版本的示例  在Visual Studio 2022上编译时需要将\*.vcxproj中“<PlatformToolset>v142</PlatformToolset>”替换为“<PlatformToolset>v143</PlatformToolset>” |  |

\*实验相关示例代码也可以从睿客网下载

链接：<https://rec.ustc.edu.cn/share/089d4b60-4bfd-11ef-a003-293c134b1311>

密码：eeis@ustc

本实验项目示例代码用到的主要函数说明如下。

1． sndPlaySound("ding.wav",SND\_SYNC);

2． PlaySound("chord.wav",NULL,SND\_SYNC );

3． 使用两个高级MCI函数

1. 播放MIDI文件示例

{

HWND hwnd;

MCI\_OPEN\_PARMS mciopen;

MCI\_OPEN\_PARMS mciplay;

DWORD rtrn;

char b[80];

hwnd=GetActiveWindow()->m\_hWnd;

mciopen.lpstrElementName = "e:\\test\\test.rmi";

mciopen.lpstrDeviceType = "sequencer" ;

rtrn = mciSendCommand(0,MCI\_OPEN,MCI\_OPEN\_TYPE|MCI\_OPEN\_ELEMENT,

(DWORD)(LPVOID)&mciopen);

if(rtrn != 0){

mciGetErrorString(rtrn,(LPSTR)b,80);

::MessageBox(hwnd,b,"MCI ERROR!",MB\_OK);

}

wDeviceID = mciopen.wDeviceID;

mciplay.dwCallback = (DWORD)hwnd;

rtrn = mciSendCommand(wDeviceID,MCI\_PLAY,MCI\_NOTIFY,(DWORD)(LPVOID)&mciplay);

if (rtrn != 0L){

mciGetErrorString(rtrn,(LPSTR)b,80);

::MessageBox(hwnd,b,"MCI Error",MB\_OK);

mciSendCommand(wDeviceID,MCI\_CLOSE,0,NULL);

}

}

//停止播放

{

HWND hwnd;

MCI\_GENERIC\_PARMS mcistop;

hwnd=GetActiveWindow()->m\_hWnd;

mcistop.dwCallback = (DWORD)hwnd;

mciSendCommand(MCI\_DEVTYPE\_SEQUENCER,MCI\_STOP,0,(DWORD)(LPVOID)&mcistop);

}

1. 录音示例

{

HWND hwnd;

MCI\_OPEN\_PARMS mciopen;

MCI\_RECORD\_PARMS mci1;

MCI\_SAVE\_PARMS mcisave;

DWORD rtrn;

char b[80];

hwnd=GetActiveWindow()->m\_hWnd;

mciopen.lpstrElementName = "e:\\ding.wav";

mcisave.lpfilename = "e:\\ding.wav";

mciopen.lpstrDeviceType = "waveaudio";

rtrn = mciSendCommand(0,MCI\_OPEN,MCI\_OPEN\_TYPE|MCI\_OPEN\_ELEMENT,

(DWORD)(LPVOID)&mciopen);

if(rtrn != 0){

mciGetErrorString(rtrn,(LPSTR)b,80);

::MessageBox(hwnd,b,"MCI ERROR!",MB\_OK);

}

wDeviceID = mciopen.wDeviceID;

mci1.dwCallback = (DWORD)hwnd;

rtrn = mciSendCommand(wDeviceID,MCI\_RECORD,MCI\_NOTIFY,(DWORD)(LPVOID)&mci1);

if (rtrn != 0L){

mciGetErrorString(rtrn,(LPSTR)b,80);

::MessageBox(hwnd,b,"MCI Error",MB\_OK);

mciSendCommand(wDeviceID,MCI\_CLOSE,0,NULL);

}

}

//保存到文件并停止录音

{

HWND hwnd;

MCI\_SAVE\_PARMS mcisave;

MCI\_OPEN\_PARMS mciopen;

DWORD rtrn;

char b[80];

hwnd=GetActiveWindow()->m\_hWnd;

mciopen.lpstrElementName = "e:\\ding.wav";

mcisave.lpfilename = "e:\\ding.wav";

mciopen.lpstrDeviceType = "waveaudio";

mcisave.dwCallback = (DWORD)hwnd;

rtrn = mciSendCommand(wDeviceID,MCI\_SAVE,MCI\_NOTIFY,(DWORD)(LPVOID)&mcisave);

if (rtrn != 0L){

mciGetErrorString(rtrn,(LPSTR)b,80);

::MessageBox(hwnd,b,"MCI Error save",MB\_OK);

mciSendCommand(wDeviceID,MCI\_CLOSE,0,NULL);

}

}

## 思考题

[1-9] 什么是MCI接口？简述MCI所囊括的内容。

[1-10] Windows下播放波形文件可以采用几种方法，各有什么优缺点？

[1-11] 在打开设备时，如下2行代码分别在WIN11操作系统和WIN7操作系统下测试通过，请查阅Microsoft Learn网站，解释为何在传递（指向 MCI\_OPEN\_PARMS 结构的）指针时，需要使用不同类型的指针。

rtrn = mciSendCommand(NULL, MCI\_OPEN, MCI\_OPEN\_TYPE | MCI\_OPEN\_ELEMENT, (DWORD\_PTR)&mciopen);

rtrn = mciSendCommand(NULL, MCI\_OPEN, MCI\_OPEN\_TYPE | MCI\_OPEN\_ELEMENT, (DWORD)(LPVOID)&mciopen);

[1-12] 查阅Microsoft Learn网站，简析用来取代MCI的MediaPlayer具备哪些功能。

# WAVE文件格式分析

## 实验目的

了解Wave文件格式并学习Windows下用低级函数来播放Wave文件。

## 预备知识

VisualC++5/6或Visual Studio2019/2022编程方法、MCI编程方法。可参阅“10 附录：Visual Studio图形界面程序开发”和“11 附录：Visual C++下的多媒体开发”。

## 实验原理

### Windows中的低级多媒体函数

Windows中的低级多媒体函数都是以mmio开头的，在Microsoft Learn的联机文档（<https://learn.microsoft.com/en-us/windows/win32/api/mmiscapi/> ）中，可以找到以下函数的具体含义和使用方法。

* mmioAdvance
* mmioAscend
* mmioClose
* mmioCreateChunk
* mmioDescend
* mmioFlush
* mmioGetInfo
* mmioInstallIOProc
* mmioOpen
* mmioRead
* mmioRename
* mmioSeek
* mmioSendMessage
* mmioSetBuffer
* mmioSetInfo
* mmioStringToFOURCC
* mmioWrite

事实上实验示例中只用到5个函数：mmioAscend、mmioClose、mmioOpen、mmioRead、mmioDescend。

### Wave文件的结构

Wave文件的数据以几种方式出现：抽样速率的各种综合、多种声道（单声和立体声）和不同分频率（每个样本的位数）。这些信息以一种格式块的形式出现-在RIFF专门名词中称为（chunk）-位于每个WAVE文件的开头附近。图 2‑1所示是一个RIFF文件的典型结构。

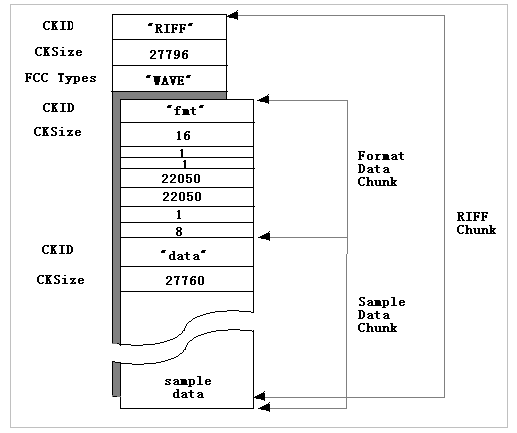


图 2‑1 RIFF文件结构示意

一个WAVE文件最少包含3个块，RIFF块是其中最大的。整个WAVE文件就是一个RIFF块。Cksize紧跟在“RIFF”CKID之后出现，它包含一个值，等于文件大小减去8个字节-这8个字节用来存储RIFF块的CKID和CKSIZE。第二和第三块称为子块，包含在RIFF块之中。这些子块的第一个是“fmt”块，包含PCMWAVEFORMAT结构所需要的信息；第二个字块“data”块是文件的最大部分，紧跟在“fmt”块后，包含所有的数据波形。RIFF块的CKSIZE等于“fmt”块和“data”块占用的字节之和。

为了读写RIFF文件，用户使用为多媒体块信息准备的叫MMCKINFO的标准数据结构。在VC中，这个结构定义为：

typedef struct {

FOURCC ckid;

DWORD cksize;

FOURCC fccType;

DWORD dwDataOffset;

DWORD dwFlags;

} MMCKINFO;

## 示例代码说明

|  |  |
| --- | --- |
| VC6版本的示例 |  |
| Visual Studio 2019版本的示例  在Visual Studio 2022上编译时需要将\*.vcxproj中“<PlatformToolset>v142</PlatformToolset>”替换为“<PlatformToolset>v143</PlatformToolset>” |  |

\*实验相关示例代码也可以从睿客网下载

链接：<https://rec.ustc.edu.cn/share/089d4b60-4bfd-11ef-a003-293c134b1311>

密码：eeis@ustc

实验示例的主要步骤为：

MMCKINFO MMCkInfoParent;

MMCKINFO MMCkInfoChild;

//打开一个WAVE文件

HMMIO hmmio = mmioOpen(FileNameAndPath.GetBuffer(80),NULL,MMIO\_READ);

//定父块，就是RIFF块的位置

MMCkInfoParent.fccType = mmioFOURCC('W','A','V','E');

mmioDescend(hmmio,&MMCkInfoParent,NULL,MMIO\_FINDRIFF);

//读如格式块内容

MMCkInfoChild.ckid = mmioFOURCC('f','m','t',' ');

mmioDescend(hmmio,&MMCkInfoChild,&MMCkInfoParent,MMIO\_FINDCHUNK);

mmioRead(hmmio,(LPSTR)&PCMWaveFmtRecord,MMCkInfoChild.cksize);

HWAVEOUT hWaveOut;

errorCode = waveOutOpen(

&hWaveOut,

WAVE\_MAPPER,

(WAVEFORMATEX\*)&PCMWaveFmtRecord,

0l,

0l,

WAVE\_FORMAT\_QUERY);

mmioAscend(hmmio,&MMCkInfoChild,0);

//读如WAVE数据块，并返回所分配内存的指针

MMCkInfoChild.ckid = mmioFOURCC('d','a','t','a');

mmioDescend(hmmio,&MMCkInfoChild,&MMCkInfoParent,MMIO\_FINDCHUNK);

long lDataSize = MMCkInfoChild.cksize;

HANDLE waveDataBlock = ::GlobalAlloc(GMEM\_MOVEABLE,lDataSize);

LPBYTE pWave = (LPBYTE)::GlobalLock(waveDataBlock);

WaveHeader.lpData = (LPSTR)pWave;

WaveHeader.dwBufferLength = lDataSize;

WaveHeader.dwFlags = 0L;

WaveHeader.dwLoops = 0L;

mmioClose(hmmio,0);

return waveDataBlock;

这样，我们从RIFF文件中抽取了数据并创建了三个基本数据结构（格式记录、波音频数据、WAVE数据头）。从内存中播放波音频需要5个步骤：打开波音频设备；生成WAVE文件头；将数据写入设备；取消WAVE文件头；关闭设备。

HWAVEOUT hWaveOut;

MMRESULT ReturnCode = waveOutOpen(

&hWaveOut,

WAVE\_MAPPER,

(WAVEFORMATEX\*)&PCMWaveFmtRecord,

0l,

0l,

0l);

ReturnCode = waveOutPrepareHeader(

hWaveOut,

&WaveHeader,

sizeof(WaveHeader));

ReturnCode = waveOutWrite(

hWaveOut,

&WaveHeader,

sizeof(WaveHeader));

do{}

while(!(WaveHeader.dwFlags & WHDR\_DONE));

ReturnCode = waveOutUnprepareHeader(hWaveOut,

&WaveHeader,

sizeof(WaveHeader));

WaveHeader.dwFlags = 0l;

ReturnCode = waveOutClose(hWaveOut);

## 实验要求

[2-1] 阅读实验原理部分，最好能够提前了解一下WAVE文件的格式。

[2-2] 参考给出的示例程序思路编制自己的程序播放WAVE文件。

[2-3] 查阅Microsoft Learn网站，了解Windows支持的MCI设备类型。

<https://learn.microsoft.com/en-us/windows/win32/multimedia/mci-device-types>

[2-4]\* [选做]编制自己的程序播放WAVE文件，可实现从任意位置开始播放（而不是从头播放），能暂停播放、继续播放、停止播放。

## 思考题

[2-5] WAVE文件保存有那些信息，以什么格式存储？

[2-6] 在同一个程序中能不能同时进行播放和录制两项操作？为什么？这对声卡有什么要求？

[2-7] 资源交换文件格式（RIFF）有什么特点？

# 基于WindowsAPI的视频捕捉

## 实验目的

学习在Windows下利用WindowsAPI进行视频捕捉卡开发的基本原理和方法。

## 预备知识

VisualC++5/6或Visual Studio2019/2022编程方法、MCI编程方法。可参阅“10 附录：Visual Studio图形界面程序开发”和“11 附录：Visual C++下的多媒体开发”。

## 实验原理

### 视频捕捉卡/摄像头编程

在DOS操作系统下，视频捕捉卡的编程是依赖于厂商提供的开发包（通常以\*.LIB文件形式出现）。在Windows操作系统下，既可以利用厂商提供开发包中的动态链接库进行程序开发，又可以利用WindowsAPI进行开发。由于厂商提供的开发包使得程序不具有兼容性，在本实验中，只介绍利用WindowsAPI进行开发的方法。

在windows3.x中，是没有视频支持的，后来微软开发了一个名称为“Video For Windows”（VFW）的视频支持产品，提供想要显示视频剪辑的Windows应用程序所需要的基本视频音频服务。Video For Windows已经被微软作为Windows操作系统的一个部分。可以利用一些API函数来访问视频设备，此时需要在程序中连接*头文件“vfw.h”*，并且将*库vfw32.lib*连接到应用程序中。

### 视频捕获有关的结构体和API

**CAPTUREPARMS 结构体**

该结构体定义在捕捉过程中视频流的有关格式参数。

typedef struct {

DWORD dwRequestMicroSecPerFrame;

BOOL fMakeUserHitOKToCapture;

UINT wPercentDropForError;

BOOL fYield;

DWORD dwIndexSize;

UINT wChunkGranularity;

BOOL fUsingDOSMemory;

UINT wNumVideoRequested;

BOOL fCaptureAudio;

UINT wNumAudioRequested;

UINT vKeyAbort;

BOOL fAbortLeftMouse;

BOOL fAbortRightMouse;

BOOL fLimitEnabled;

UINT wTimeLimit;

BOOL fMCIControl;

BOOL fStepMCIDevice;

DWORD dwMCIStartTime;

DWORD dwMCIStopTime;

BOOL fStepCaptureAt2x;

UINT wStepCaptureAverageFrames;

DWORD dwAudioBufferSize;

BOOL fDisableWriteCache;

UINT AVStreamMaster;

} CAPTUREPARMS;

**函数capCreateCaptureWindow（）**

调用格式：

HWND VFWAPI capCreateCaptureWindow( *//功能：创建一个捕获窗口。*

LPCSTR lpszWindowName, *//捕获窗口的名称*

DWORD dwStyle, *//捕获窗口的窗口风格*

int x, int y, *//捕获窗口的左上角坐标*

int nWidth, int nHeight, *//捕获窗口的的宽和高*

HWND hWnd, *//捕获窗口的的父窗口句柄*

int nID *//捕获窗口的的ID值*

);

使用示例：

HWND m\_hCapture;

m\_hCapture = capCreateCaptureWindow ("Cap",WS\_EX\_CONTROLPARENT

| WS\_CHILD | WS\_VISIBLE, 4, 4, 320, 240, m\_hWnd, 0);

**函数capDriverConnect（）**

连接视频捕获装置到捕获窗口

capDriverConnect (m\_hCapture, 0);

**函数capCaptureGetSetup（）**

返回视频流的有关参数设置

CAPTUREPARMS m\_CaptureParms;

capCaptureGetSetup (m\_hCapture, &m\_CaptureParms, sizeof (m\_CaptureParms));

**函数capOverlay（）**

使得当前窗口为Overlay窗口，在窗口上显示摄入视频信息。

capOverlay (m\_hCapture, TRUE);

**函数 capGrabFrame（）**

捕获当前的一贞画面到捕获窗口

capGrabFrame(m\_hCapture);

**函数 capFileSaveDIB（）**

将当前捕获到的一贞画面保存到文件

capFileSaveDIB(m\_hCapture,"e:\\user\\test.bmp");

**函数capFileSaveAs（）**

将当前捕获到的动画保存到文件

capFileSaveAs( m\_hCapture,"e:\\user\\test.avi");

**函数 capDlgVideoCompression（）**

显示系统的视频压缩格式对话框

capDlgVideoCompression(m\_hCapture);

**函数 capDlgVideoSource（）**

显示视频捕捉卡的驱动程序设置对话框

capDlgVideoSource(m\_hCapture);

（注意：此处仅列出在示例程序中使用到的API函数，其他的API函数请参阅Microsoft Learn 联机资料。）

## 实验要求

[3-1] 参考示例程序，学习视频设备编程的基本方法，掌握基于Windows API进行视频捕获的基本流程。示例程序无法显示视频时，检查代码中关于分辨率的设置。

[3-2] 查阅Microsoft Learn网站，了解LPVIDEOHDR结构体各成员变量的含义。

<https://learn.microsoft.com/en-us/windows/win32/api/vfw/ns-vfw-videohdr>

[3-3] 查阅Microsoft Learn网站，了解vfw.h所定义的函数（functions）。

<https://learn.microsoft.com/en-us/windows/win32/api/vfw/>

[3-4] 自己制作一个程序，显示由摄像头摄入的影像。

[3-5] 修改自己的程序，使之能够将抓取的图象和动画保存到硬盘上。

## 示例代码说明

|  |  |
| --- | --- |
| VC6版本的示例 |  |
| Visual Studio 2019版本的示例  在Visual Studio 2022上编译时需要将\*.vcxproj中“<PlatformToolset>v142</PlatformToolset>”替换为“<PlatformToolset>v143</PlatformToolset>” |  |

\*实验相关示例代码也可以从睿客网下载

链接：<https://rec.ustc.edu.cn/share/089d4b60-4bfd-11ef-a003-293c134b1311>

密码：eeis@ustc

示例程序“开始捕捉”按钮启动视频捕获。

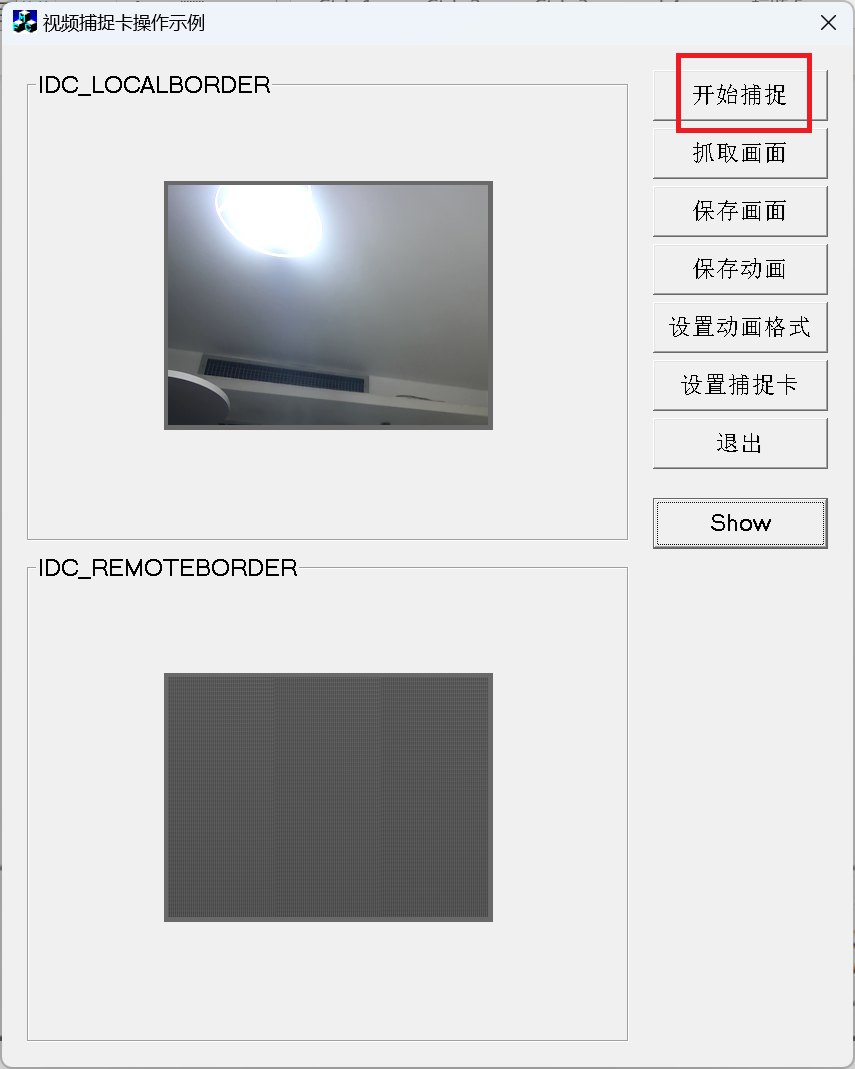


图 3‑1 视频捕获示例程序界面示意

在开始捕获的时候，需要选择所用PC上连接的视频捕捉设备（视频捕捉卡、或者USB摄像头），类似下图。

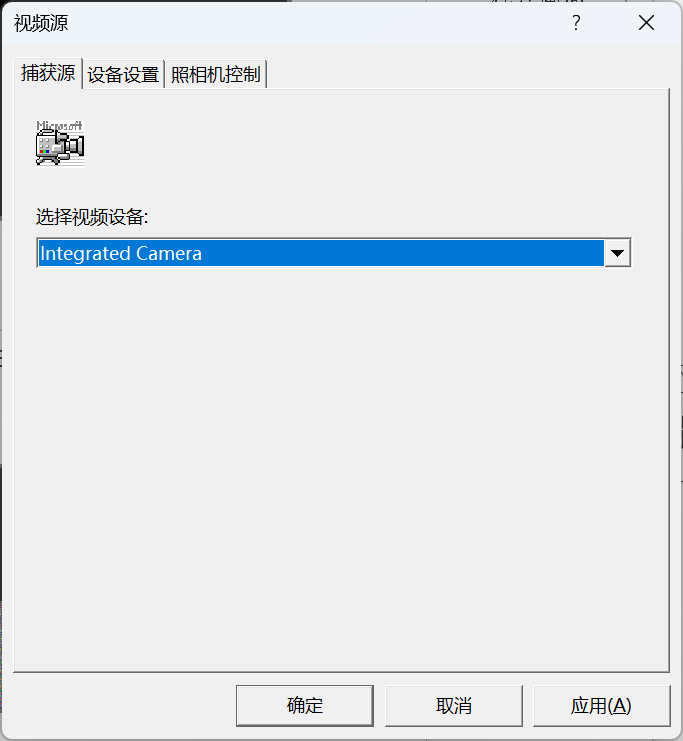
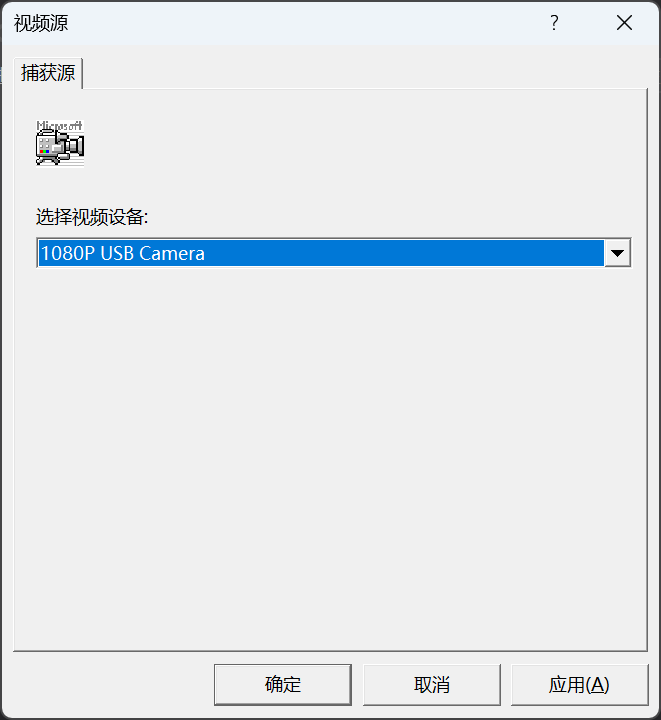
 

图 3‑2 Windows下捕获源设置界面示意

## 思考题

[3-6] Video For Windows提供一些什么功能？请归类列举。

[3-7] 简析回调函数（Callback Function）的作用。

# JPEG静图像压缩

## 实验目的

1 了解数字图像压缩的一类方法。

2 掌握2D快速DCT变换的算法。

## 实验原理

JPEG 标准是面向连续色调静止图像的图像压缩标准。它定义了多种类型的 工作模式，其中最基本的是基于8×8块的 DCT 变换的顺序编码。就是将一帧图像分为8×8的数据块单元，按照从左至右、自上而下的顺序对“块流”编码，其编码，解码的框图如下：



图 4‑1 基于DCT的编码、解码框图

## 实验内容

按照图 4‑1所示框图实现一帧图像的压缩编码和解码。

### DCT 变换

表达式为：





其中：

 其它

二维 DCT 变换从表达式看具有可分离性，即可以分解为行和列的一维DCT 变换的组合运算。另一种 2D 快速余弦变换是把8×8的块分成更小的子块，直接对二维数据块操作。

参照教材，实现2D 快速余弦变换的正变换和反变换，并将它和分解为行和列的一维运算的算法的进行比较。

### 量化

量化的表达式为：



其中 g\_scale 是量化因子。对 DC 系统，g\_scale 恒为8；对 AC 系数，它可以是1～30 的整数。是上式四舍五入取整的结果，称为量化系数。Q（U,V）是量化矩阵。

按照表1.所给的量化矩阵对 DCT 系统进行量化（所有的表在附录四“JPEG 数字图像压缩方法”中）。

### 熵编码

a. DC 系统的编码。

DC系统编码的基本步骤为：

1）以初始值为128，对相邻块的DC 系统作差分：DIFFj=DCj-DCj-1;

2）将差分值改写为〔size，value〕(〔尺寸，幅值〕)的符号对。



3）按照表2，3对该符号对编码。

b. AC 系数的编码。

AC系数编码的基本步骤为：

1）以块为单位，从 AC01 开始，对块内的量化数据进行Z 字形行程扫描。记下两T 非零值的大小和其间连续零的个数。

2）将扫描结果改写为〔runlength，level〕(〔行程长，幅值〕)的符号对；其中 runlength 表示该非零值和它前一个值之间零的个数，以 EOB 标识一个块的扫描结果。

3）按照表3.对上面的符号对编码。若符号对在表中查不到对应码字时，以 ESC 码表示，然后按表4，5 对 runlength 和 level 编码。

4. 对编码后的图像数据进行解码。

## 实验要求

[4-1] 请按照实验内容部分的要求完成自己的编解码程序（可以利用示例程序中的辅助程序进行检查）。

## 示例代码

实验示例代码：<http://staff.ustc.edu.cn/~network/mmt/JPEG_sample_code20220108.rar>

\*实验相关示例代码也可以从睿客网下载

链接：<https://rec.ustc.edu.cn/share/089d4b60-4bfd-11ef-a003-293c134b1311>

密码：eeis@ustc

实验示例使用说明：解压附件后，打开"jpeg\_r"文件夹中的"jpeg\_r.sln"。运行后可以在与 "jpeg\_r.sln"处于同一文件夹中的"jpeg\_r"文件夹中找到编码输出文件 " lady\_pre.dat "。 " lady\_pre.dat "文件中存放的就是编码之后的结果（未加JPEG头）。 只需将 " lady\_pre.dat "文件加JPEG 头即可得到 JPEG图像。具体做法是将 " lady\_pre.dat "文件拷贝到" JPEG编码 "文件夹下的" jpeg\_r\_followup "文件夹，运行 " GENJPG.exe"， 按照要求 从键盘输入 " lady\_pre.dat "和"\*\*\*.jpg"即可。

以下是示例程序介绍。

### 示例程序的JPEG图像编码过程

输入源图象

输出图象

8\*8二维DCT变换

量化

8\*8二维IDCT变换

逆量化

量化表

直流分量预测编码

直流分量预测解码

2. 熵编码

1.表说明

熵解码

图 4‑2 JPEG编码及解码原理图

### 示例程序文件介绍

实验室给出以下用BC开发的程序（JPEG目录下，包括原代码）：

（1）JPEG压缩程序gendat.exe；

（2）为压缩后的数据加上JPEG文件头的程序genjpg.exe；

（3）解JPEG压缩程序extrdat.exe。

其中（1）的输入参数是需要生成的压缩文件的名字，主程序流程为：

a.程序初始化，读取文件名称、原始图象等基本数据；

b.从AC.TAB和DC.TAB两个文件生成AC表和DC表，存入数组；

c.以8×8为样本单位进行DCT变换；

d.以8×8为样本单位进行量化；

e.以8×8为样本单位进行Z型扫描；

f.以8×8为样本单位进行编码；

g.以8×8为样本单位进行解量化；

h.以8×8为样本单位进行IDCT变换；

请同学们在参考上述程序的基础上，编制自己的JPEG压缩程序，然后利用2和3检查自己的程序是否正确运行（注意：\*.JPG文件可以用浏览器IE查看）。

## 思考题

[4-2] 计算图像的压缩比，并比较原图像的效果。

[4-3] 改变 g\_scale 和量化矩阵中的元素，比较压缩比和恢复图像的效果。

[4-4] DCT 变换，量化后，数据的值域是多少？有没有可能超出码表所给幅值的范围？

# Android下的音频编程

## 实验目的

* 了解RIFF文件格式。
* 了解Android对媒体回放、录制的支持库。
* 学习Android下进行声音播放的基本方法。
* 学习Android下进行声音录制的基本方法。

## 预备知识

Android Studio编程方法。可参阅“12 附录：Android Studio程序开发极简入门”。

## 实验原理

### Android APIs对多媒体的支持

android.media包下面包含了Android开发中媒体类，可以支持各类媒体的回放、录制。部分和声音、图像、视频有关的类定义如表 1所示。

得益于在android.media中定义的丰富函数，Android下可以采用多种方式来进行声音的播放或录制[1]。例如，（1）**MediaPlayer**支持AAC、AMR、FLAC、MP3、MIDI、OGG、PCM等格式，MediaPlayer可以通过设置元数据和播放源来音频。（2）**SoundPool**支持多个音频文件同时播放(组合音频也是有上限的)，延时短，比较适合短促、密集的场景。（3）**AudioTrack**属于更偏底层的音频播放。可用于单个音频播放和管理，相比于MediaPlayer具有：精炼、高效的优点。也可以用于播放PCM(PCM无压缩的音频格式)音乐流的回放；如果要播需放其它格式音频，需要相应的解码器。

在Android中录音可以用MediaRecord录音。AudioRecorder 录音声音数据的编码格式为PCM格式，但PCM语音数据，直接保存成音频文件，是不能够被播放器播放的，所以需要实现PCM语音数据转为WAV文件。

表 1 android.media中与媒体控制有关的部分类[2]

|  |  |
| --- | --- |
|  |  |
| [AsyncPlayer](http://www.android-doc.com/reference/android/media/AsyncPlayer.html) | Plays a series of audio URIs, but does all the hard work on another thread so that any slowness with preparing or loading doesn't block the calling thread. |
| [AudioFormat](http://www.android-doc.com/reference/android/media/AudioFormat.html) | The AudioFormat class is used to access a number of audio format and channel configuration constants. |
| [AudioManager](http://www.android-doc.com/reference/android/media/AudioManager.html) | AudioManager provides access to volume and ringer mode control. |
| [AudioRecord](http://www.android-doc.com/reference/android/media/AudioRecord.html) | The AudioRecord class manages the audio resources for Java applications to record audio from the audio input hardware of the platform. |
| [AudioTimestamp](http://www.android-doc.com/reference/android/media/AudioTimestamp.html) | Structure that groups a position in frame units relative to an assumed audio stream, together with the estimated time when that frame was presented or is committed to be presented. |
| [AudioTrack](http://www.android-doc.com/reference/android/media/AudioTrack.html) | The AudioTrack class manages and plays a single audio resource for Java applications. |
| [FaceDetector](http://www.android-doc.com/reference/android/media/FaceDetector.html) | Identifies the faces of people in a [Bitmap](http://www.android-doc.com/reference/android/graphics/Bitmap.html) graphic object. |
| [FaceDetector.Face](http://www.android-doc.com/reference/android/media/FaceDetector.Face.html) | A Face contains all the information identifying the location of a face in a bitmap. |
| [Image](http://www.android-doc.com/reference/android/media/Image.html) | A single complete image buffer to use with a media source such as a [MediaCodec](http://www.android-doc.com/reference/android/media/MediaCodec.html). |
| [ImageReader](http://www.android-doc.com/reference/android/media/ImageReader.html) | The ImageReader class allows direct application access to image data rendered into a [Surface](http://www.android-doc.com/reference/android/view/Surface.html)  Several Android media API classes accept Surface objects as targets to render to, including [MediaPlayer](http://www.android-doc.com/reference/android/media/MediaPlayer.html), [MediaCodec](http://www.android-doc.com/reference/android/media/MediaCodec.html), and [RenderScript Allocations](http://www.android-doc.com/reference/android/renderscript/Allocation.html). |
| [MediaCodec](http://www.android-doc.com/reference/android/media/MediaCodec.html) | MediaCodec class can be used to access low-level media codec, i.e. |
| [MediaCrypto](http://www.android-doc.com/reference/android/media/MediaCrypto.html) | MediaCrypto class can be used in conjunction with [MediaCodec](http://www.android-doc.com/reference/android/media/MediaCodec.html) to decode encrypted media data. |
| [MediaFormat](http://www.android-doc.com/reference/android/media/MediaFormat.html) | Encapsulates the information describing the format of media data, be it audio or video. |
| [MediaMuxer](http://www.android-doc.com/reference/android/media/MediaMuxer.html) | MediaMuxer facilitates muxing elementary streams. |
| [MediaRecorder](http://www.android-doc.com/reference/android/media/MediaRecorder.html) | Used to record audio and video. |
| [MediaRecorder.AudioEncoder](http://www.android-doc.com/reference/android/media/MediaRecorder.AudioEncoder.html) | Defines the audio encoding. |
| [MediaRecorder.AudioSource](http://www.android-doc.com/reference/android/media/MediaRecorder.AudioSource.html) | Defines the audio source. |
| [MediaRecorder.OutputFormat](http://www.android-doc.com/reference/android/media/MediaRecorder.OutputFormat.html) | Defines the output format. |
| [MediaRecorder.VideoEncoder](http://www.android-doc.com/reference/android/media/MediaRecorder.VideoEncoder.html) | Defines the video encoding. |
| [MediaRecorder.VideoSource](http://www.android-doc.com/reference/android/media/MediaRecorder.VideoSource.html) | Defines the video source. |
| [MediaRouter](http://www.android-doc.com/reference/android/media/MediaRouter.html) | MediaRouter allows applications to control the routing of media channels and streams from the current device to external speakers and destination devices. |
| [Ringtone](http://www.android-doc.com/reference/android/media/Ringtone.html) | Ringtone provides a quick method for playing a ringtone, notification, or other similar types of sounds. |
| [SoundPool](http://www.android-doc.com/reference/android/media/SoundPool.html) | The SoundPool class manages and plays audio resources for applications. |

### WAVE文件格式

WAVE(\*.WAV文件)是Microsoft开发的一种音频文件格式，它符合RIFF文件格式标准。RIFF全称为资源互换文件格式（Resources Interchange File Format），是Windows下大部分多媒体文件遵循的一种文件结构。

根据RIFF规范，其基本的组成单元是chunk。[3]一个WAV文件通常有三个chunk以及一个可选chunk，其在文件中的排列方式依次是：RIFF chunk，Format chunk，Fact chunk（附加块，可选），Data chunk。如图 5‑1所示。

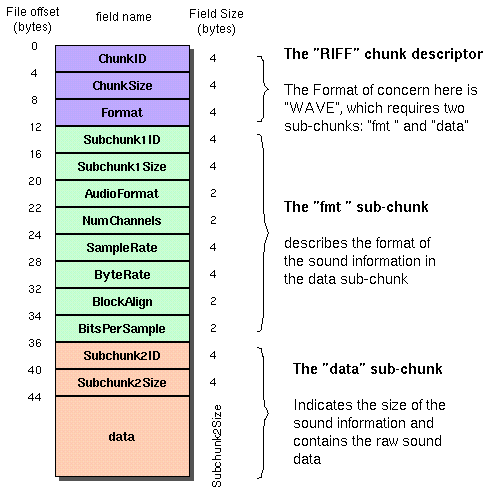


图 5‑1 \*.WAV文件的RIFF格式示意[4]

### android.media声音播放的几个类

<http://www.android-doc.com/reference/android/media/MediaPlayer.html>

<http://www.android-doc.com/reference/android/media/SoundPool.html>

<http://www.android-doc.com/reference/android/media/AudioTrack.html>

### android.meida录制声音的类

<http://www.android-doc.com/reference/android/media/AudioRecord.html>

使用Android设备的麦克风录音需要设备访问权限在AndroidManifest.xml 文件中的对应描述如下[[1]](#footnote-1)。

 <uses-permission android:name="android.permission.RECORD\_AUDIO" />

实际上，本示例中播放声音的代码访问了手机SDCard中存放的文件，还需要存储设备的读取权限，在AndroidManifest.xml 文件中的对应描述如下。

 <uses-permission android:name=" android.permission.READ\_EXTERNAL\_STORAGE " />

## 示例说明

2018年版本实验示例代码：<http://staff.ustc.edu.cn/~network/mmt/AudioSample.rar>

注20241006：由于Android SDK版本更新，2018年的示例代码的编译环境（基于Gradle 4.4）搭建较为困难，故抽取核心代码后再新Android Studio版本（Android Studio Koala Feature Drop | 2024.1.2 Build，Gradle 8.7）下重新设计了示例。新示例将声音播放和录音两部分功能分至2个不同的例子：**5\_AudioPlayer-20241006.rar**、**5\_AudioRecorder-20241006.rar**。鉴于Goolgle目前主推用Kotlin来写gradle，java语法的build.gradle改为build.gradle.kts。

\*实验相关示例代码也可以从睿客网下载

链接：<https://rec.ustc.edu.cn/share/089d4b60-4bfd-11ef-a003-293c134b1311>

密码：eeis@ustc

本实验项目所提供示例中用到的主要函数如下所示。

|  |  |
| --- | --- |
| **函数名称** | **函数功能** |
| **void** play\_with\_media\_palyer(String str\_file\_name) | 使用MediaPlayer播放本地音乐文件 |
| **void** play\_with\_sound\_pool(String str\_file\_name) | 使用SoundPool播放本地音乐文件 |
| **void** play\_with\_audio\_track(String str\_file\_name) | 使用AudioTrack播放本地音乐文件 |
| **boolean** makePCMFileToWAVFile(String pcmPath, String destinationPath, **boolean** deletePcmFile) | 将PCM样本文件添加WAV文件的文件头 |
| **void** audio\_record\_5s(String str\_file\_name) | 使用AudioRecord进行录音的简单示例 |
| **void** start\_stop\_audio\_record() | 使用多线程进行录音控制的的简单示例 |

### 示例运行效果

图 5‑2 示例界面效果（左：2018版APP，中：AudioPlayer，右：AudioRecorder）

2024年版AudioRecorder录音文件的保存位置：

\Android\data\cn.edu.ustc.eeis.audiorecorder\files\\*

对应的，在虚拟机中，文件的保存位置：

\storage\emulated\0\data\cn.edu.ustc.eeis.audiorecorder\files\\*

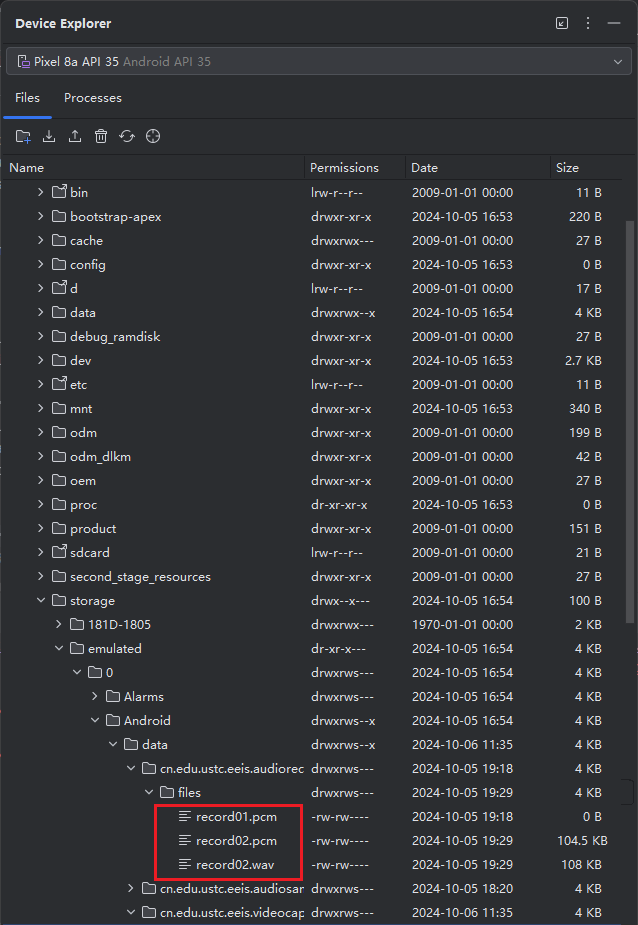


图 5‑3 AudioRecorder示例在虚拟机中录音文件的保存路径

## 实验要求

[5-1] 根据实验原理部分所列出的信息做进一步的文献调研；详细学习WAVE文件格式。找一个WAV文件（如C:\Windows\media\Ring01.wav），采取合适的方法分析其声道数目、采样频率、样本精度。结果写入实验报告。

[5-2] 编译链接所给AudioSample示例，阅读声音播放的三个函数，做必要的修改，播放手机特定目录下的声音文件。请将关键代码写入实验报告。

[5-3] 阅读声音录制的2个函数，做必要的修改，自己进行声音的录制。示例代码给出了多线程实现的一种方式。请进行进一步的文献调研，学习多线程代码有哪些不同的实现方式。相关结论请记录到实验报告中。

## 思考题

[5-4] 单声道的声音保存为\*.WAV文件和多声道的声音保存成\*.WAV文件后应该在文件的哪个部分标识声道有关的信息？

[5-5] 请通过实验验证你所用的手机（或其他Android设备）在同一个APP中能否同时有多个声音播放软件同时播放不同的声音文件？

[5-6] 请通过实验验证你所用的手机（或其他Android设备）能否在同一个APP中同时进行声音的播放和录制？

## 参考文献

[1] CNBLOGS. 【Android】播放音频的几种方式介绍 [Online] Available: <https://www.cnblogs.com/HDK2016/p/8043247.html>

[2] AndroidAPIs. android.media [Online] Available: <http://www.android-doc.com/reference/android/media/package-summary.html>

[3] CNBLOGS. RIFF和WAVE音频文件格式 [Online] Available: <https://www.cnblogs.com/wangguchangqing/p/5957531.html>

[4] Unknown. WAVE PCM soundfile format [Online] Available: <http://soundfile.sapp.org/doc/WaveFormat/>

# Android下的摄像头编程

## 实验目的

* 学习Android下进行摄像头控制的基本方法。
* 理解回调函数。

## 实验原理

### android.hardware

android.hardware包下面包含了如摄像头、传感器等硬件的支持（不是所有的硬件）。如表 1所示。

表 2 android.hardware中类[1]

|  |  |
| --- | --- |
| **类** | **类的功能** |
| [Camera](http://www.android-doc.com/reference/android/hardware/Camera.html) | The Camera class is used to set image capture settings, start/stop preview, snap pictures, and retrieve frames for encoding for video. |
| [Camera.Area](http://www.android-doc.com/reference/android/hardware/Camera.Area.html) | The Area class is used for choosing specific metering and focus areas for the camera to use when calculating auto-exposure, auto-white balance, and auto-focus. |
| [Camera.CameraInfo](http://www.android-doc.com/reference/android/hardware/Camera.CameraInfo.html) | Information about a camera |
| [Camera.Face](http://www.android-doc.com/reference/android/hardware/Camera.Face.html) | Information about a face identified through camera face detection. |
| [Camera.Parameters](http://www.android-doc.com/reference/android/hardware/Camera.Parameters.html) | Camera service settings. |
| [Camera.Size](http://www.android-doc.com/reference/android/hardware/Camera.Size.html) | Image size (width and height dimensions). |
| [ConsumerIrManager](http://www.android-doc.com/reference/android/hardware/ConsumerIrManager.html) | Class that operates consumer infrared on the device. |
| [ConsumerIrManager.CarrierFrequencyRange](http://www.android-doc.com/reference/android/hardware/ConsumerIrManager.CarrierFrequencyRange.html) | Represents a range of carrier frequencies (inclusive) on which the infrared transmitter can transmit |
| [GeomagneticField](http://www.android-doc.com/reference/android/hardware/GeomagneticField.html) | Estimates magnetic field at a given point on Earth, and in particular, to compute the magnetic declination from true north. |
| [Sensor](http://www.android-doc.com/reference/android/hardware/Sensor.html) | Class representing a sensor. |
| [SensorEvent](http://www.android-doc.com/reference/android/hardware/SensorEvent.html) | This class represents a [Sensor](http://www.android-doc.com/reference/android/hardware/Sensor.html) event and holds informations such as the sensor's type, the time-stamp, accuracy and of course the sensor's [data](http://www.android-doc.com/reference/android/hardware/SensorEvent.html#values). |
| [SensorManager](http://www.android-doc.com/reference/android/hardware/SensorManager.html) | SensorManager lets you access the device's [sensors](http://www.android-doc.com/reference/android/hardware/Sensor.html). |
| [TriggerEvent](http://www.android-doc.com/reference/android/hardware/TriggerEvent.html) | This class represents a Trigger Event - the event associated with a Trigger Sensor. |
| [TriggerEventListener](http://www.android-doc.com/reference/android/hardware/TriggerEventListener.html) | This class is the listener used to handle Trigger Sensors. |

### android.hardware.Camera类

现在的手机一般都会提供相机功能，有些相机的镜头甚至支持1000万以上像素，有些甚至支持光学变焦，这些手机已经变成了专业数码相机。为了充分利用手机上的相机功能，Android APIs提供了丰富的支持库。如表 3所示。

使用Android设备的摄像头（照相机）需要设备访问权限在AndroidManifest.xml 文件中的对应描述如下[[2]](#footnote-2)。

 <uses-permission android:name="android.permission.CAMERA" />  
 <uses-feature android:name="android.hardware.camera" />  
 <uses-feature android:name="android.hardware.camera.autofocus" />

表 3 android.hardware.Camera的成员函数

|  |  |
| --- | --- |
|  | **成员函数名称** |
| final void | [addCallbackBuffer](http://www.android-doc.com/reference/android/hardware/Camera.html#addCallbackBuffer(byte[]))(byte[] callbackBuffer)  Adds a pre-allocated buffer to the preview callback buffer queue. |
| final void | [autoFocus](http://www.android-doc.com/reference/android/hardware/Camera.html#autoFocus(android.hardware.Camera.AutoFocusCallback))([Camera.AutoFocusCallback](http://www.android-doc.com/reference/android/hardware/Camera.AutoFocusCallback.html) cb)  Starts camera auto-focus and registers a callback function to run when the camera is focused. |
| final void | [cancelAutoFocus](http://www.android-doc.com/reference/android/hardware/Camera.html#cancelAutoFocus())()  Cancels any auto-focus function in progress. |
| final boolean | [enableShutterSound](http://www.android-doc.com/reference/android/hardware/Camera.html#enableShutterSound(boolean))(boolean enabled)  Enable or disable the default shutter sound when taking a picture. |
| static void | [getCameraInfo](http://www.android-doc.com/reference/android/hardware/Camera.html#getCameraInfo(int, android.hardware.Camera.CameraInfo))(int cameraId, [Camera.CameraInfo](http://www.android-doc.com/reference/android/hardware/Camera.CameraInfo.html) cameraInfo)  Returns the information about a particular camera. |
| static int | [getNumberOfCameras](http://www.android-doc.com/reference/android/hardware/Camera.html#getNumberOfCameras())()  Returns the number of physical cameras available on this device. |
| [Camera.Parameters](http://www.android-doc.com/reference/android/hardware/Camera.Parameters.html) | [getParameters](http://www.android-doc.com/reference/android/hardware/Camera.html#getParameters())()  Returns the current settings for this Camera service. |
| final void | [lock](http://www.android-doc.com/reference/android/hardware/Camera.html#lock())()  Re-locks the camera to prevent other processes from accessing it. |
| static [Camera](http://www.android-doc.com/reference/android/hardware/Camera.html) | [open](http://www.android-doc.com/reference/android/hardware/Camera.html#open(int))(int cameraId)  Creates a new Camera object to access a particular hardware camera. |
| static [Camera](http://www.android-doc.com/reference/android/hardware/Camera.html) | [open](http://www.android-doc.com/reference/android/hardware/Camera.html#open())()  Creates a new Camera object to access the first back-facing camera on the device. |
| final void | [reconnect](http://www.android-doc.com/reference/android/hardware/Camera.html#reconnect())()  Reconnects to the camera service after another process used it. |
| final void | [release](http://www.android-doc.com/reference/android/hardware/Camera.html#release())()  Disconnects and releases the Camera object resources. |
| void | [setAutoFocusMoveCallback](http://www.android-doc.com/reference/android/hardware/Camera.html#setAutoFocusMoveCallback(android.hardware.Camera.AutoFocusMoveCallback))([Camera.AutoFocusMoveCallback](http://www.android-doc.com/reference/android/hardware/Camera.AutoFocusMoveCallback.html) cb)  Sets camera auto-focus move callback. |
| final void | [setDisplayOrientation](http://www.android-doc.com/reference/android/hardware/Camera.html#setDisplayOrientation(int))(int degrees)  Set the clockwise rotation of preview display in degrees. |
| final void | [setErrorCallback](http://www.android-doc.com/reference/android/hardware/Camera.html#setErrorCallback(android.hardware.Camera.ErrorCallback))([Camera.ErrorCallback](http://www.android-doc.com/reference/android/hardware/Camera.ErrorCallback.html) cb)  Registers a callback to be invoked when an error occurs. |
| final void | [setFaceDetectionListener](http://www.android-doc.com/reference/android/hardware/Camera.html#setFaceDetectionListener(android.hardware.Camera.FaceDetectionListener))([Camera.FaceDetectionListener](http://www.android-doc.com/reference/android/hardware/Camera.FaceDetectionListener.html) listener)  Registers a listener to be notified about the faces detected in the preview frame. |
| final void | [setOneShotPreviewCallback](http://www.android-doc.com/reference/android/hardware/Camera.html#setOneShotPreviewCallback(android.hardware.Camera.PreviewCallback))([Camera.PreviewCallback](http://www.android-doc.com/reference/android/hardware/Camera.PreviewCallback.html) cb)  Installs a callback to be invoked for the next preview frame in addition to displaying it on the screen. |
| void | [setParameters](http://www.android-doc.com/reference/android/hardware/Camera.html#setParameters(android.hardware.Camera.Parameters))([Camera.Parameters](http://www.android-doc.com/reference/android/hardware/Camera.Parameters.html) params)  Changes the settings for this Camera service. |
| final void | [setPreviewCallback](http://www.android-doc.com/reference/android/hardware/Camera.html#setPreviewCallback(android.hardware.Camera.PreviewCallback))([Camera.PreviewCallback](http://www.android-doc.com/reference/android/hardware/Camera.PreviewCallback.html) cb)  Installs a callback to be invoked for every preview frame in addition to displaying them on the screen. |
| final void | [setPreviewCallbackWithBuffer](http://www.android-doc.com/reference/android/hardware/Camera.html#setPreviewCallbackWithBuffer(android.hardware.Camera.PreviewCallback))([Camera.PreviewCallback](http://www.android-doc.com/reference/android/hardware/Camera.PreviewCallback.html) cb)  Installs a callback to be invoked for every preview frame, using buffers supplied with [addCallbackBuffer(byte[])](http://www.android-doc.com/reference/android/hardware/Camera.html#addCallbackBuffer(byte[])), in addition to displaying them on the screen. |
| final void | [setPreviewDisplay](http://www.android-doc.com/reference/android/hardware/Camera.html#setPreviewDisplay(android.view.SurfaceHolder))([SurfaceHolder](http://www.android-doc.com/reference/android/view/SurfaceHolder.html) holder)  Sets the [Surface](http://www.android-doc.com/reference/android/view/Surface.html) to be used for live preview. |
| final void | [setPreviewTexture](http://www.android-doc.com/reference/android/hardware/Camera.html#setPreviewTexture(android.graphics.SurfaceTexture))([SurfaceTexture](http://www.android-doc.com/reference/android/graphics/SurfaceTexture.html) surfaceTexture)  Sets the [SurfaceTexture](http://www.android-doc.com/reference/android/graphics/SurfaceTexture.html) to be used for live preview. |
| final void | [setZoomChangeListener](http://www.android-doc.com/reference/android/hardware/Camera.html#setZoomChangeListener(android.hardware.Camera.OnZoomChangeListener))([Camera.OnZoomChangeListener](http://www.android-doc.com/reference/android/hardware/Camera.OnZoomChangeListener.html) listener)  Registers a listener to be notified when the zoom value is updated by the camera driver during smooth zoom. |
| final void | [startFaceDetection](http://www.android-doc.com/reference/android/hardware/Camera.html#startFaceDetection())()  Starts the face detection. |
| final void | [startPreview](http://www.android-doc.com/reference/android/hardware/Camera.html#startPreview())()  Starts capturing and drawing preview frames to the screen. |
| final void | [startSmoothZoom](http://www.android-doc.com/reference/android/hardware/Camera.html#startSmoothZoom(int))(int value)  Zooms to the requested value smoothly. |
| final void | [stopFaceDetection](http://www.android-doc.com/reference/android/hardware/Camera.html#stopFaceDetection())()  Stops the face detection. |
| final void | [stopPreview](http://www.android-doc.com/reference/android/hardware/Camera.html#stopPreview())()  Stops capturing and drawing preview frames to the surface, and resets the camera for a future call to [startPreview()](http://www.android-doc.com/reference/android/hardware/Camera.html#startPreview()). |
| final void | [stopSmoothZoom](http://www.android-doc.com/reference/android/hardware/Camera.html#stopSmoothZoom())()  Stops the smooth zoom. |
| final void | [takePicture](http://www.android-doc.com/reference/android/hardware/Camera.html#takePicture(android.hardware.Camera.ShutterCallback, android.hardware.Camera.PictureCallback, android.hardware.Camera.PictureCallback))([Camera.ShutterCallback](http://www.android-doc.com/reference/android/hardware/Camera.ShutterCallback.html) shutter, [Camera.PictureCallback](http://www.android-doc.com/reference/android/hardware/Camera.PictureCallback.html) raw, [Camera.PictureCallback](http://www.android-doc.com/reference/android/hardware/Camera.PictureCallback.html) jpeg)  Equivalent to takePicture(shutter, raw, null, jpeg). |
| final void | [takePicture](http://www.android-doc.com/reference/android/hardware/Camera.html#takePicture(android.hardware.Camera.ShutterCallback, android.hardware.Camera.PictureCallback, android.hardware.Camera.PictureCallback, android.hardware.Camera.PictureCallback))([Camera.ShutterCallback](http://www.android-doc.com/reference/android/hardware/Camera.ShutterCallback.html) shutter, [Camera.PictureCallback](http://www.android-doc.com/reference/android/hardware/Camera.PictureCallback.html) raw, [Camera.PictureCallback](http://www.android-doc.com/reference/android/hardware/Camera.PictureCallback.html) postview, [Camera.PictureCallback](http://www.android-doc.com/reference/android/hardware/Camera.PictureCallback.html) jpeg)  Triggers an asynchronous image capture. |
| final void | [unlock](http://www.android-doc.com/reference/android/hardware/Camera.html#unlock())()  Unlocks the camera to allow another process to access it. |

### Camera2类说明

Android 5.0(SDK 21)之后android.hardware.Camera就被废弃了，取而代之的是全新的android.hardware.Camera2 。Android 5.0对拍照API进行了全新的设计，新增了全新设计的Camera v2 API，这些API不仅大幅提高了Android系统拍照的功能，还能支持RAW照片输出，甚至允许程序调整相机的对焦模式、曝光模式、快门等[2]。

* CameraManager：摄像头管理器。这是一个全新的系统管理器，专门用于检测系统摄像头、打开系统摄像头。除此之外，调用CameraManager的getCameraCharacteristics(String)方法即可获取指定摄像头的相关特性。
* CameraCharacteristics：摄像头特性。该对象通过CameraManager来获取，用于描述特定摄像头所支持的各种特性。
* CameraDevice：代表系统摄像头。该类的功能类似于早期的Camera类。
* CameraCaptureSession：这是一个非常重要的API，当程序需要预览、拍照时，都需要先通过该类的实例创建Session。而且不管预览还是拍照，也都是由该对象的方法进行控制的，其中控制预览的方法为setRepeatingRequest()；控制拍照的方法为capture()。
* CameraRequest和CameraRequest.Builder：当程序调用setRepeatingRequest()方法进行预览时，或调用capture()方法进行拍照时，都需要传入CameraRequest参数。CameraRequest代表了一次捕获请求，用于描述捕获图片的各种参数设置，比如对焦模式、曝光模式……总之，程序需要对照片所做的各种控制，都通过CameraRequest参数进行设置。CameraRequest.Builder则负责生成CameraRequest对象。

### android.hardware.ImageReader类

与播放声音不同，在使用摄像头的时候，我们需要把摄像头捕获的图像进行显示。示例程序中使用ImageReader类完成摄像头捕获的图像数据到界面上显示控件的显示。类ImageReader支持图像数据的着色（The ImageReader class allows direct application access to image data rendered into a Surface）。

表 4 android.hardware.ImageReader类的成员函数[3]

|  |  |
| --- | --- |
| **函数返回值类型** | **成员函数名称** |
| [Image](http://www.android-doc.com/reference/android/media/Image.html) | [acquireLatestImage](http://www.android-doc.com/reference/android/media/ImageReader.html#acquireLatestImage())()  Acquire the latest [Image](http://www.android-doc.com/reference/android/media/Image.html) from the ImageReader's queue, dropping older [images](http://www.android-doc.com/reference/android/media/Image.html). |
| [Image](http://www.android-doc.com/reference/android/media/Image.html) | [acquireNextImage](http://www.android-doc.com/reference/android/media/ImageReader.html#acquireNextImage())()  Acquire the next Image from the ImageReader's queue. |
| void | [close](http://www.android-doc.com/reference/android/media/ImageReader.html#close())()  Free up all the resources associated with this ImageReader. |
| int | [getHeight](http://www.android-doc.com/reference/android/media/ImageReader.html#getHeight())()  The height of each [Image](http://www.android-doc.com/reference/android/media/Image.html), in pixels. |
| int | [getImageFormat](http://www.android-doc.com/reference/android/media/ImageReader.html#getImageFormat())()  The [image format](http://www.android-doc.com/reference/android/graphics/ImageFormat.html) of each Image. |
| int | [getMaxImages](http://www.android-doc.com/reference/android/media/ImageReader.html#getMaxImages())()  Maximum number of images that can be acquired from the ImageReader by any time (for example, with [acquireNextImage()](http://www.android-doc.com/reference/android/media/ImageReader.html#acquireNextImage())). |
| [Surface](http://www.android-doc.com/reference/android/view/Surface.html) | [getSurface](http://www.android-doc.com/reference/android/media/ImageReader.html#getSurface())()  Get a [Surface](http://www.android-doc.com/reference/android/view/Surface.html) that can be used to produce [Images](http://www.android-doc.com/reference/android/media/Image.html) for this ImageReader. |
| int | [getWidth](http://www.android-doc.com/reference/android/media/ImageReader.html#getWidth())()  The width of each [Image](http://www.android-doc.com/reference/android/media/Image.html), in pixels. |
| static [ImageReader](http://www.android-doc.com/reference/android/media/ImageReader.html) | [newInstance](http://www.android-doc.com/reference/android/media/ImageReader.html#newInstance(int, int, int, int))(int width, int height, int format, int maxImages)  Create a new reader for images of the desired size and format. |
| void | [setOnImageAvailableListener](http://www.android-doc.com/reference/android/media/ImageReader.html#setOnImageAvailableListener(android.media.ImageReader.OnImageAvailableListener, android.os.Handler))([ImageReader.OnImageAvailableListener](http://www.android-doc.com/reference/android/media/ImageReader.OnImageAvailableListener.html) listener, [Handler](http://www.android-doc.com/reference/android/os/Handler.html) handler)  Register a listener to be invoked when a new image becomes available from the ImageReader. |

表 5 android.hardware.ImageReader类的回调接口[3]

|  |  |  |
| --- | --- | --- |
| interface | [ImageReader.OnImageAvailableListener](http://www.android-doc.com/reference/android/media/ImageReader.OnImageAvailableListener.html) | Callback interface for being notified that a new image is available. |

### 回调函数

回调函数就是一个通过函数指针调用的函数。如果你把函数的指针（地址）作为参数传递给另一个函数，当这个指针被用来调用其所指向的函数时，我们就说这是回调函数。回调函数不是由该函数的实现方直接调用，而是在特定的事件或条件发生时由另外的一方调用的，用于对该事件或条件进行响应。

多媒体操作往往需要回调函数的支持，例如，我们通过摄像头捕获图像，启动摄像头后，会通知摄像头数据采集有关的函数，等采集完一帧图像后，你要做一些处理（回调函数中就是这些处理的具体代码）。

## 示例说明

实验示例代码：<http://staff.ustc.edu.cn/~network/mmt/VideoCapture.rar>

注20241006：由于Android SDK版本更新，2018年的示例代码的编译环境（基于Gradle 4.4）搭建较为困难，故抽取核心代码后在新Android Studio版本（Android Studio Koala Feature Drop | 2024.1.2 Build，Gradle 8.7）下重新设计了示例：**6\_VideoCapture-20241006.rar**。鉴于Goolgle目前主推用Kotlin来写gradle，java语法的build.gradle改为build.gradle.kts。

\*实验相关示例代码也可以从睿客网下载

链接：<https://rec.ustc.edu.cn/share/089d4b60-4bfd-11ef-a003-293c134b1311>

密码：eeis@ustc

示例代码用到的主要的类如下。

**private** CameraManager **mCameraManager**;  
*\* The ImageReader class allows direct application access to image data rendered into a Surface \*/***private** ImageReader **mImageReader**;  
*/\*系统向摄像头发送 Capture 请求，而摄像头会返回 CameraMetadata，这一切都是在由对应的CameraDevice创建的CameraCaptureSession 会话完成，当程序需要预览、拍照、再次预览时，都需要先通过会话\*/***private** CameraCaptureSession **mCameraCaptureSession**;  
*/\*CameraDevice是Camera2中抽象出来的一个对象，直接与系统硬件摄像头相联系\*/***private** CameraDevice **mCameraDevice**;  
*/\*CameraRequest代表了一次捕获请求，而CameraRequest.Builder用于描述捕获图片的各种参数设置，包含捕获硬件（传感器，镜头，闪存），对焦模式、曝光模式，处理流水线，控制算法和输出缓冲区的配置\*/***private** CaptureRequest.Builder **pictureBuilder**;

### 示例运行效果

2024年版VideoCapture拍照后图片文件的保存位置：

\Android\data\cn.edu.ustc.eeis.videocapture\files\\*

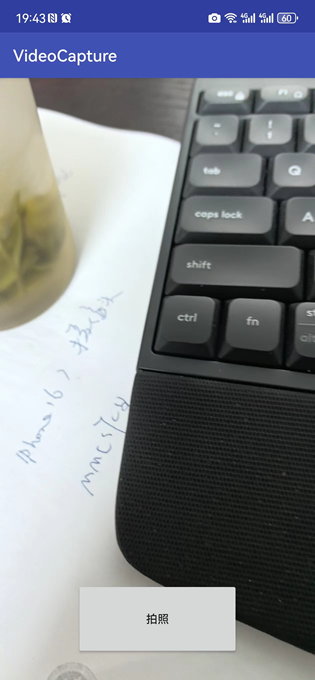
 

图 6‑1 照相机拍照示例效果（左图：APP界面，右图：虚拟机下抓拍文件的保存目录）

## 实验要求

[6-1] 编译示例程序，运行结果截图放入实验报告。

[6-2] 读懂示例程序VideoCapture的操作步骤，在实验报告中画出访问摄像头的流程示意图。

## 思考题

[6-3] 示例代码中用到的TextureView与一般的View有什么不同？

[6-4] 阐述一下个人关于Callback函数用途的理解。

## 参考文献

[1] AndroidAPIs. android.hardware [Online] Available: <http://www.android-doc.com/reference/android/hardware/package-summary.html>

[2] 简书. Android Camera2 使用总结 [Online] Available: <https://www.jianshu.com/p/73fed068a795>

[3] AndroidAPIs. ImageReader [Online] Available: <http://www.android-doc.com/reference/android/media/ImageReader.html>

# Android下使用Tensorflow模型

## 实验目的

* 了解Tensorflow模型文件的生成和保存。
* 学习Android下使用C++库的基本步骤。
* 学习Android下使用 Tensorflow所生成的模型。

## 实验原理

### Python开发环境Anaconda

Anaconda指的是一个开源的Python发行版本，其包含了conda、Python等180多个科学包及其依赖项。因为包含了大量的科学包，Anaconda 的下载文件比较大，如果只需要某些包，或者需要节省带宽或存储空间，也可以使用Miniconda。

Anaconda包括Conda、Python以及一大堆安装好的工具包，比如：numpy、pandas等。其中Conda是一个开源的包、环境管理器，可以用于在同一个机器上安装不同版本的软件包及其依赖，并能够在不同的环境之间切换。Miniconda包括Conda、Python。

### Python下使用Tensorflow

MNIST是一个简单的计算机视觉数据集，它包含手写数字的图像集。可以从这个简单的数据集开始练习Tensorflow的使用。Tensorflow编程的一个基本步骤大致为：1、定义数据；2、定义计算图与变量；3、定义会话；4、进行计算。建议阅读[1]学习在MNIST上进行神经网络模型搭建的简单Tensorflow例程。

### Android JNI

JNI的全称就是Java Native Interface，顾名思义，就是Java和C/C++相互通信的接口。可以通过JNI 调用系统提供的 API。操作系统，无论是 Linux，Windows 还是 Mac OS，或者一些汇编语言写的底层硬件驱动都是 C/C++ 写的。Java和C/C++不同 ，它不会直接编译成平台机器码，而是编译成虚拟机可以运行的Java字节码的.class文件，通过JIT技术即时编译成本地机器码，所以有效率就比不上C/C++代码，JNI技术就解决了这一痛点，JNI 可以说是 C 语言和 Java 语言交流的适配器、中间件[2]。

Android提供了3个实用的函数用来加载JNI库，分别是System.loadLibrary(libname)，Runtime.getRuntime().loadLibrary(libname)，以及Runtime.getRuntime().load(libFilePath)[3]。用System.loadLibrary(libname)和Runtime.getRuntime().loadLibrary(libname)这两个函数加载so库，不需要指定so库的路径，Android会默认从系统的共享库目录里面去查找，Android的共享库目录就是vendor/lib和system/lib，如果在共享库路径里面找到指定名字的so库，就会立即加载这个so库，所以我们给so库起名的时候要尽量避免和Android共享库里面的so库同名。如果在共享库目录里面查找不到，就会在APP的安装目录里面查找APP的私有so库，如果查找到，会立即加载这个so库。

Android Studio通过CMakeLists.txt文件配置需要生成的so库。CMakeLists.txt文件中使用add\_library函数用来配置要生成的so库的基本信息，比如库的名字，要生成的so库是静态的还是共享的，so库的C/C++源文件列表。使用find\_library函数来从NDK目录下面查找特定的so库。使用target\_link\_libraries函数用来把要生成的so库和依赖的其它so库进行链接，生成我们需要的so库文件。在app目录下新建CmakeLists.txt文件，其实要是在Android Studio新建项目时勾选了使用c++，那么这个文件和一些其他配置已经有了，稍微修改就行。

### 在Android上集成Tensorflow

谷歌开源库Tensorflow可被用在安卓系统中实现机器学习。Tensorflow的核心是用C++编写的；为了在安卓上搭建Tensorflow，我们需要用JNI（Java本地接口）来调用C++函数，比如说loadModel，getPredictions，等等。

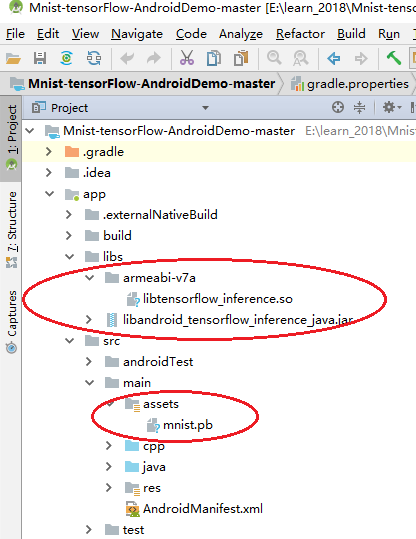


图 7‑1 Android上运行Tensorflow需要的库及模型文件示意

准备好Tensorflow支持库和已经在Tensorflow下生成的模型文件，如图 7‑1所示。按照如下步骤在将有关库文件放入Android项目[4, 5]。

1. 新建一个Android Project。
2. 把刚才的pb文件存放到assets文件夹下。
3. 将libandroid\_tensorflow\_inference\_java.jar存放到/app/libs目录下，并且右键“add as Libary”。
4. 在/app/libs下新建armeabi文件夹，并将libtensorflow\_inference.so放进去。

配置app:gradle以及gradle.properties。

在android节点下添加soureSets，用于制定jniLibs的路径

sourceSets {

main {

jniLibs.srcDirs = ['libs']

}

}12345

在defaultConfig节点下添加

defaultConfig {

ndk {

abiFilters "armeabi"

}

}123456

在gradle.properties中添加下面一行

android.useDeprecatedNdk=true1

### 后续Android下对Tensorflow的支持

根据Google的官方说明[6]，手机上的Tensorflow支持从2019年开始从TensorFlow Mobile向TensorFlow Lite迁移。

在Android下使用Tensorflow lite模型的方法可以参阅，

Android 快速入门  |  TensorFlow (google.cn)

<https://tensorflow.google.cn/lite/android/quickstart?hl=bg>

官方提供的Android Studio下使用Tensorflow lite模型的示例可以参阅，这些例子build之后可以直接在手机上测试效果。

GitHub - tensorflow/examples: TensorFlow examples

<https://github.com/tensorflow/examples/>

笔者注：由于Android Studio及相关软件工具的版本升级都很快，往往遇到示例无法正常build的情况。多数时候时因为学习者和示例设计人员的软件配置环境不同导致，使用Tensoflow官网示例前请阅读“12.2.5 进阶：Android Studio各相关软件包的兼容性”。

## 实验原理补充更新2024

前述示例各个步骤的验证在ARMv7a架构的手机和虚拟机上完成的。在高版本的Android Studio（Android Studio Koala Feature Drop | 2024.1.2）下重新设计程序编译后在ARMv8a架构的手机上无法正常调用Tensorflow库，识别图片后闪退。检查代码后删除其中用C代码实现的一个整数转字符串函数。闪退问题应当和仅仅支持ARMv7a架构没有关系，但是没有找到合适ARMv8a架构版本的\*.so库（及加载库的\*.jar），示例代码仍仅支持ARMv7a架构。

## 示例说明

实验示例代码：<http://staff.ustc.edu.cn/~network/mmt/android_tensorflow.rar>

注20241006：由于Android SDK版本更新，2018年的示例代码的编译环境（基于Gradle 4.4）搭建较为困难，故抽取核心代码后在新Android Studio版本（Android Studio Koala Feature Drop | 2024.1.2 Build，Gradle 8.7）下重新设计了示例：**7\_TensorflowAndroid-20241007.rar**。鉴于Goolgle目前主推用Kotlin来写gradle，java语法的build.gradle改为build.gradle.kts。

\*实验相关示例代码也可以从睿客网下载

链接：<https://rec.ustc.edu.cn/share/089d4b60-4bfd-11ef-a003-293c134b1311>

密码：eeis@ustc

本实验项目所提供的示例代码参考[5]进行局部修改和增删，主要文件的功能如表 6所示。示例代码详细说明参见文件中注释信息和原始博文[5]。

表 6 示例代码功能说明

|  |  |
| --- | --- |
| **文件名称** | **功能说明** |
| Mnist-generate-tensorFlow-model\mnist\_test.py | 基于MNIST的Tensorflow简单例子 |
| Mnist-generate-tensorFlow-model\mnist-prediction.py | 基于MNIST训练一个模型 |
| Mnist-generate-tensorFlow-model\mnist-train.py | 利用训练好的模型进行手写体识别 |
| Mnist-tensorFlow-AndroidDemo-master目录 | 集成了Tensorflow支持库的Android示例项目 |

## 实验要求

### Python+Tensorflow环境配置

[7-1] 安装anaconda，设置好环境，详细步骤可参考[7]。

[7-2] 安装Tensorflow，详细步骤可参考[8]。

### Tensorflow例程运行与模型生成

[7-3] 阅读并运行Mnist-generate-tensorFlow-model子目录中的mnist\_test.py。该文件是实现FC卷积神经网络识别mnist手写体的简单例子，参考[1]。记录运行结果中识别的正确率。

[7-4] 阅读并运行Mnist-generate-tensorFlow-model子目录中的另外两个Python文件。其中mnist-prediction.py利用.py利用TensorFlow生成MNIST手写数字识别的模型文件（缺省存放为model\mnist.pb）；mnist-prediction.py利用已经生成的模型文件对测试图片进行数字识别。

### 将Tensorflow下生成的模型集成到Android项目中

[7-5] 将上一步骤中生成的模型文件拷贝到Android项目Mnist-tensorFlow-AndroidDemo-master对应的子目录中（Mnist-tensorFlow-AndroidDemo-master\app\src\main\assets\ mnist.pb）；编译Mnist-tensorFlow-AndroidDemo-master项目。

[7-6] 将上一步骤中生成的APK文件在自己的Android手机上进行测试，类似图 7‑2的界面效果。



图 7‑2 2018版APK运行界面示意

## 思考题

[7-7] 做扩展文献调研，分析TensorFlow下生成的模型文件包含哪些信息？

[7-8] Android下JNI的作用是什么？

[7-9] Armeabi-v7a指的是什么？

## 参考文献

[1] CSDN. python深度学习库tensorflow——实现FC卷积神经网络识别mnist手写体 [Online] Available: <https://blog.csdn.net/luanpeng825485697/article/details/79100008>

[2] 简书. Android NDK开发：JNI基础篇 [Online] Available: <https://www.jianshu.com/p/ac00d59993aa>

[3] CSDN. Android JNI基础篇 [Online] Available: <https://blog.csdn.net/kgdwbb/article/details/72810251>

[4] CSDN. 将tensorflow训练好的模型移植到android上 [Online] Available: <https://blog.csdn.net/cxq234843654/article/details/71171293>

[5] CSDN. 将tensorflow训练好的模型移植到Android (MNIST手写数字识别) [Online] Available: <https://blog.csdn.net/guyuealian/article/details/79672257>

[6] Google. Building TensorFlow on Android [Online] Available: <https://www.tensorflow.org/lite/tfmobile/android_build>

[7] JobBole. Anaconda使用总结 [Online] Available: <http://python.jobbole.com/86236/>

[8] CSDN. 手把手教你如何安装Tensorflow（Windows和Linux两种版本） [Online] Available: <https://blog.csdn.net/cs_hnu_scw/article/details/79695347>

# SDL+ffmpeg播放mp4

## 实验目的

* 学习Visual Studio中SDL编程的流程。
* 了解ffmpeg工具包作用。
* 学习ffmpeg工具包中命令行的使用。
* 学习Visual Studio中基于ffmpeg库编程的流程。

## 实验原理

### SDL

SDL（Simple DirectMedia Layer）是一套开放源代码的跨平台多媒体开发库，使用C语言写成。SDL提供了数种控制图像、声音、输出入的函数，让开发者只要用相同或是相似的代码就可以开发出跨多个平台（Linux、Windows、Mac OS X等）的应用软件。目前SDL多用于开发游戏、模拟器、媒体播放器等多媒体应用领域。

官网：<http://libsdl.org/>

官方文档：<http://wiki.libsdl.org/FrontPage>

在SDL2.0中，提供了如下不同类别的函数[1]：Basics、Video、Input Events、Force Feedback、Audio、Threads、Timers、File Abstraction、Shared Object Support、Platform and CPU Information、Power Management、Additional Functionality。

### ffmpeg

FFmpeg是一个开源免费跨平台的视频和音频流方案，属于自由软件，采用LGPL或GPL许可证。它提供了录制、转换以及流化音视频的完整解决方案。FFmpeg在Linux平台下开发，但它同样也可以在其它操作系统环境中编译运行，包括Windows、Mac OS X等。FFmpeg有非常强大的功能包括视频采集功能、视频格式转换、视频抓图、给视频加水印等。

FFmpeg有三种版本。**静态库（static）版本**里面有3个exe： ffmpeg.exe， ffplay.exe， ffprobe.exe，每个exe体积很大，因为相关的dll已经被编译到exe 里面去了。作为工具而言此版本就可以满足一般应用的需求。**动态库（shared）版本**，里面有3个exe：ffmpeg.exe，ffplay.exe，ffprobe.exe，还有一些dll。Shared版本里面的exe体积很小，这些exe在运行的时候，到相应的dll中调用功能。程序运行过程必须依赖于提供的dll文件。**开发者（dev）版本**，是用于开发的，里面包含了库文件 xxx.lib 以及头文件 xxx.h，这个版本不包含exe文件。

## 示例说明

|  |  |
| --- | --- |
| 示例代码 |  |
| 开发包等 | <https://rec.ustc.edu.cn/share/fec97780-11b5-11eb-bb2f-854bc28a6a11> |

实验示例代码：<http://staff.ustc.edu.cn/~network/mmt/SDL_ffmpeg_player.rar>

实验示例代码配套开发包：https://rec.ustc.edu.cn/share/fec97780-11b5-11eb-bb2f-854bc28a6a11

\*实验相关示例代码也可以从睿客网下载

链接：<https://rec.ustc.edu.cn/share/089d4b60-4bfd-11ef-a003-293c134b1311>

密码：eeis@ustc

实验示例由[2]例子改编而来（如下嵌入文件）。在Visual Studio 2017环境测试通过。代码中需要阅读的源文件是SDL\_ffmpeg\_player.cpp，使用前请阅读源代码中注释信息。

示例代码编译连接前需要修改项目属性中：

* 头文件包含路径，含SDL和ffmpeg
* 库文件包含路径，含SDL和ffmpeg
* 需要输入的库文件，\*.lib

另外需要修改代码中：拟播放的mp4文件名称。需要的MP4测试文件，自行下载，如可从以下链接下载：

http://clips.vorwaerts-gmbh.de/big\_buck\_bunny.mp4

http://vjs.zencdn.net/v/oceans.mp4

https://media.w3.org/2010/05/sintel/trailer.mp4

http://mirror.aarnet.edu.au/pub/TED-talks/

示例代码生成EXE后，需要拷贝有关的DLL至EXE文件所在目录。具体需要拷贝的DLL名称参见上级内容部分说明。

## 上机内容

### SDL环境配置

[8-1] 下载SDL软件包

从<http://libsdl.org/>下载Development Libraries中VC对应的版本。实验示例编写时（2020.10.19）最新版本为[SDL2-devel-2.0.12-VC.zip](http://libsdl.org/release/SDL2-devel-2.0.12-VC.zip) (Visual C++ 32/64-bit)。下载后的软件包解压后备用。

[8-2] 阅读SDL文档[3]。

[8-3] Visual Studio下测试SDL

依据[4]教程，在Visual Studio下新建一个项目（Console类型的项目），设置项目属性。需要配置：

* Setting the Include Path，例如E:\project\_2020\SDL2-2.0.12\include;
* Setting the Library Path，例如E:\project\_2020\SDL2-2.0.12\lib;
* Adding the Library Dependencies， SDL2.lib; SDL2main.lib;

[8-4] 测试如下代码，

#include <iostream>

#include <SDL.h>

int main(int, char\*\*){

**if** (SDL\_Init(SDL\_INIT\_VIDEO) != 0){

std::cout << "SDL\_Init Error: " << SDL\_GetError() << std::endl;

**return** 1;

}

SDL\_Quit();

**return** 0;

}

注意：生成EXE后，运行前需要把SDL2.DLL拷贝到和EXE文件相同的目录下。并且，依据Visual Studio中配置是x64还是x86，要拷贝对应的DLL版本。

### ffmpeg环境测试

[8-5] 下载ffmpeg。目前官网（https://ffmpeg.org/download.html）给出2个可下载Windows用版本的网址：

<https://www.gyan.dev/ffmpeg/builds/>

<https://github.com/BtbN/FFmpeg-Builds/releases>

但是最新版本调试过程有问题，实验示例调试过程采用的是ffmpeg-4.2.3-win64-dev和ffmpeg-4.2.3-win64-shared。

[8-6] 了解“FFmpeg Tools”中的三个程序的作用。阅读ffprobe帮助文档（<https://ffmpeg.org/ffprobe.html>）；阅读ffmpeg帮助文档（<https://ffmpeg.org/ffmpeg.html>）；阅读ffplay帮助文档（<https://ffmpeg.org/ffplay.html>）。

[8-7] 使用ffplay播放mp4文件。尝试播放过程中控制操作（暂停、继续、增加音量、减小音量…）；改变显示窗口的大小。

[8-8] 在使用ffmpeg前，需要配置Visual Studio，可参考[5]。在Visual Studio下新建一个项目（Console类型），弄成如下配置：

* 1. 添加头文件目录，C例如，E:\cxh\_work\FFmpeg\ffmpeg-4.2.3-win64-dev\include;
  2. 添加库文件目录，例如，E:\cxh\_work\FFmpeg\ffmpeg-4.2.3-win64-dev\lib;
  3. 添加附加依赖库，顺序为：avcodec.lib; avformat.lib; avutil.lib; avdevice.lib; avfilter.lib;postproc.lib;swresample.lib; swscale.lib;

[8-9] 测试如下代码，

#include "pch.h"

#include <iostream>

extern "C" {

#include "libavcodec/avcodec.h"

#include "libavformat/avformat.h"

}

int main(int argc, char\* argv[]) {

printf("%s\nOK!", avcodec\_configuration());

return 0;

}

注意：生成EXE后，运行前需要把几个DLL（avcodec-58.dll avdevice-58.dll avfilter-7.dll avformat-58.dll avutil-56.dll postproc-55.dll swresample-3.dll swscale-5.dll）拷贝到和EXE文件相同的目录下。并且，依据Visual Studio中配置是x64还是x86，要拷贝对应的DLL版本。

### 调试示例代码

[8-10] 将示例SDL\_ffmpeg\_player.rar解压后用Visual Studio打开项目。参考代码（SDL\_ffmpeg\_player.cpp）中注释信息，修改项目属性的头文件包含目录、库文件包含目录、指定需要输入的库文件（\*.lib）。

[8-11] 测试播放mp4文件的效果。

### 完善示例代码(\*选作)

[8-12] \*[选做]在示例代码上添加播放控制功能（如暂停、停止、视频适应窗口大小进行缩放）。

## 思考题

[8-13] 查阅SDL文档[3]，说明SDL\_CreateWindow的作用是什么？

[8-14] 示例程序只能播放连续的图像，不能播放声音。自行查询资料，如果在播放连续图像的同时还播放声音，需要做哪些扩展？

## 参考文献

[1] S. Wiki. "SDL 2.0 API by Category." <http://wiki.libsdl.org/APIByCategory> (accessed 20240725).

[2] 简书-MzDavid. "FFmpeg+SDL2实现视频流播放." <https://www.jianshu.com/p/002060731418> (accessed 20240725).

[3] S. Wiki. "Simple DirectMedia Layer: SDL Tutorials." <http://wiki.libsdl.org/> & <http://wiki.libsdl.org/Tutorials> (accessed 20240725).

[4] S. I. Will Usher, University of Utah. "TwinklebearDev SDL 2.0 Tutorial, Lesson 0: Setting up SDL." <https://www.willusher.io/pages/sdl2/> & <https://www.willusher.io/sdl2%20tutorials/2013/08/15/lesson-0-visual-studio> (accessed 20240725).

[5] cnblogs-灰色飘零. "VS 2017配置FFmpeg开发环境." <https://www.cnblogs.com/renhui/p/10387764.html> (accessed 20240725).

# 附录：JEPG参数表

**表1 量化矩阵*Q(u,v)***

**16 11 10 16 24 40 51 61**

**12 12 14 19 26 58 60 55**

**14 13 16 24 40 57 69 56**

**14 17 22 29 51 87 80 62**

**18 22 37 56 68 109 103 77**

**24 35 55 64 81 104 113 92**

**49 64 78 87 103 121 120 101**

**72 92 95 98 112 100 103 99**

**表2 DC系数表**

**位长 位数 码字**

**0 2 00**

**1 3 010**

**2 3 011**

**3 3 100**

**4 3 101**

**5 3 110**

**6 4 1110**

**7 5 11110**

**8 6 111110**

**9 7 1111110**

**10 8 11111110**

**11 9 111111110**

**表3 AC系数表**

**行程/尺寸 码长 码字**

**0/0(EOB) 4 1010**

**0/1 2 00**

**0/2 2 01**

**0/3 3 100**

**0/4 4 1011**

**0/5 5 11010**

**0/6 7 1111000**

**0/7 8 11111000**

**0/8 10 1111110110**

**0/9 16 1111111110000010**

**0/A 16 1111111110000011**

**1/1 4 1100**

**1/2 5 11011**

**1/3 7 1111001**

**1/4 9 111110110**

**1/5 11 11111110110**

**1/6 16 1111111110000100**

**1/7 16 1111111110000101**

**1/8 16 1111111110000110**

**1/9 16 1111111110000111**

**1/A 16 1111111110001000**

**2/1 5 11100**

**2/2 8 11111001**

**2/3 10 1111110111**

**2/4 12 111111110100**

**2/5 16 111111111001001**

**2/6 16 111111111001010**

**2/7 16 111111111001011**

**2/8 16 111111111001100**

**2/9 16 111111111001101**

**2/A 16 111111111001110**

**3/1 6 111010**

**3/2 9 111110111**

**3/3 12 111111110101**

**3/4 16 1111111110001111**

**3/5 16 1111111110010000**

**3/6 16 1111111110010001**

**3/7 16 1111111110010010**

**3/8 16 1111111110010011**

**3/9 16 1111111110010100**

**3/A 16 1111111110010101**

**4/1 6 111011**

**4/2 10 1111111000**

**4/3 16 1111111110010110**

**4/4 16 1111111110010111**

**4/5 16 1111111110011000**

**4/6 16 1111111110011001**

**4/7 16 1111111110011010**

**4/8 16 1111111110011011**

**4/9 16 1111111110011100**

**4/A 16 1111111110011101**

**5/1 7 1111010**

**5/2 11 11111110111**

**5/3 16 1111111110011110**

**5/4 16 1111111110011111**

**5/5 16 1111111110100000**

**5/6 16 1111111110100001**

**5/7 16 1111111110100010**

**5/8 16 1111111110100011**

**5/9 16 1111111110100100**

**5/A 16 1111111110100101**

**6/1 7 1111011**

**6/2 12 111111110110**

**6/3 16 1111111110100110**

**6/4 16 1111111110100111**

**6/5 16 1111111110101000**

**6/6 16 1111111110101001**

**6/7 16 1111111110101010**

**6/8 16 1111111110101011**

**6/9 16 1111111110100100**

**6/A 16 1111111110100101**

**7/1 8 11111010**

**7/2 12 111111110111**

**7/3 16 1111111110101110**

**7/4 16 1111111110101111**

**7/5 16 1111111110110000**

**7/6 16 1111111110110001**

**7/7 16 1111111110110010**

**7/8 16 1111111110110011**

**7/9 16 1111111110110100**

**7/A 16 1111111110110101**

**8/1 9 111111000**

**8/2 15 111111111000000**

**8/3 16 1111111110110110**

**8/4 16 1111111110110111**

**8/5 16 1111111110111000**

**8/6 16 1111111110111001**

**8/7 16 1111111110111010**

**8/8 16 1111111110111011**

**8/9 16 1111111110111100**

**8/A 16 1111111110111101**

**9/1 9 111111001**

**9/2 16 1111111110111110**

**9/3 16 1111111110111111**

**9/4 16 1111111111000000**

**9/5 16 1111111111000001**

**9/6 16 1111111111000010**

**9/7 16 1111111111000011**

**9/8 16 1111111111000100**

**9/9 16 1111111111000101**

**9/A 16 1111111111000110**

**A/1 9 111111010**

**A/2 16 1111111111000111**

**A/3 16 1111111111001000**

**A/4 16 1111111111001001**

**A/5 16 1111111111001010**

**A/6 16 1111111111001011**

**A/7 16 1111111111001100**

**A/8 16 1111111111001101**

**A/9 16 1111111111001110**

**A/A 16 1111111111001111**

**B/1 10 1111111001**

**B/2 16 1111111111010000**

**B/3 16 1111111111010001**

**B/4 16 1111111111010010**

**B/5 16 1111111111010011**

**B/6 16 1111111111010100**

**B/7 16 1111111111010101**

**B/8 16 1111111111010110**

**B/9 16 1111111111010111**

**B/A 16 1111111111011000**

**C/1 10 1111111010**

**C/2 16 1111111111011001**

**C/3 16 1111111111011010**

**C/4 16 1111111111011011**

**C/5 16 1111111111011100**

**C/6 16 1111111111011101**

**C/7 16 1111111111011110**

**C/8 16 1111111111011111**

**C/9 16 1111111111100000**

**C/A 16 1111111111100001**

**D/1 11 11111111000**

**D/2 16 1111111111100010**

**D/3 16 1111111111100011**

**D/4 16 1111111111100100**

**D/5 16 1111111111100101**

**D/6 16 1111111111100110**

**D/7 16 1111111111100111**

**D/8 16 1111111111101000**

**D/9 16 1111111111101001**

**D/A 16 1111111111101010**

**E/1 16 1111111111101011**

**E/2 16 1111111111101100**

**E/3 16 1111111111101101**

**E/4 16 1111111111101110**

**E/5 16 1111111111101111**

**E/6 16 1111111111110000**

**E/7 16 1111111111110001**

**E/8 16 1111111111110010**

**E/9 16 1111111111110011**

**E/A 16 1111111111110100**

**F/0(ZRL) 11 11111111001**

**F/1 16 1111111111110101**

**F/2 16 1111111111110110**

**F/3 16 1111111111110111**

**F/4 16 1111111111111000**

**F/5 16 1111111111111001**

**F/6 16 1111111111111010**

**F/7 16 1111111111111011**

**F/8 16 1111111111111100**

**F/9 16 1111111111111101**

**F/A 16 1111111111111110**

**注意：在AC系数编码过程中，当连续的'0'的个数大于等于16，就进行一次编码，行程/位长为15/0。**

**表4 正负值表**

|  |  |  |
| --- | --- | --- |
| **位长**  **(Size)** | **振幅**  **(Amplitude)** | **码字**  **(Code Word) (binary)** |
| **0** | **0** | **-** |
| **1** | **-1,1** | **0,1** |
| **2** | **-3,-2,2,3** | **00,01,10,11** |
| **3** | **-7..-4,4..7** | **000,…,011,100,…111** |
| **4** | **-15..-8,8..15** | **0000,…,0111,1000,…,1111** |
| **.**  **.**  **.** | **.**  **.**  **.** | **.**  **.**  **.** |
| **16** | **-32767...-16384,16384,…,32767** | **.** |

# 附录：Visual Studio图形界面程序开发

本节内容介绍Visual Studio下开发MFC图形界面程序的基本步骤。示例（如下嵌入文件）在Visual Studio 2022（VS 2022）下测试。



## 概述

### Project & Solution

Visual Studio中的解决方案和项目是两个重要的概念，它们各自扮演着不同的角色，共同构成了软件开发的基础。

* 项目（Project）：项目是Visual Studio中用于逻辑上管理、生成和调试构成应用程序的代码、资源和其他文件的单元。它包含了项目的所有文件和元数据，以及项目的构建配置和属性设置。项目是开发人员实际编写代码的地方，它包含了源代码文件、资源文件以及项目特定的设置信息。
* 解决方案（Solution）：解决方案是一个容器，用于组织一个或多个相关的代码项目。它类似于一个工作区或文件夹，可以包含多个项目，这些项目可能包括类库项目和对应的测试项目等。解决方案帮助开发者更好地管理和组织他们的代码，使得相关的项目可以在一起被构建和管理。

简而言之，项目是开发具体应用程序的单位，而解决方案则是一个包含一个或多个项目的容器，用于更好地组织和管理工作。

项目 (Visual Studio SDK) | Microsoft Learn

<https://learn.microsoft.com/zh-cn/previous-versions/bb165067(v=vs.120)>

解决方案 | Microsoft Learn

<https://learn.microsoft.com/zh-cn/previous-versions/bb165922(v=vs.120)>

解决方案是创建应用程序的分组一个或多个项。 与解决方案中的项目和状态信息在两个解决方案文件、基于文本的解决方案 (.sln) 文件和二进制解决方案用户选项 (.suo) 文件中。 这些文件类似于用于 Visual Basic的早期版本中的组 (.vbg) 文件和工作区 (.dsw) 文件和用于 Visual C++的早期版本的用户选项 (.opt) 文件。

### MFC 应用程序概述

MFC（Microsoft Fundamental Class）全称叫做微软基础类库，是微软公司开发的一个C++类库，主要封装了大部分Windows API函数。此外，MFC不仅仅是一个类库，还是一个框架，每次新建一个MFC的工程，VS会自动生成一些基础文件，封装MFC的内核，这样在编程时就不需要考虑一些底层功能的实现，程序员可以专心实现自己想要的功能。

创建 MFC 应用程序 | Microsoft Learn

<https://learn.microsoft.com/zh-cn/cpp/mfc/reference/creating-an-mfc-application?view=msvc-170&source=recommendations>

MFC 应用程序是基于 Microsoft 基础类 (MFC) 库的 Windows 可执行应用程序。 MFC 可执行程序通常分为五类：标准 Windows 应用程序、对话框、基于窗体的应用程序、资源管理器样式的应用程序和 Web 浏览器样式的应用程序。

### Visual Studio安装和配置

访问微软官网下载社区版

下载 Visual Studio Tools - 免费安装 Windows、Mac、Linux (microsoft.com)

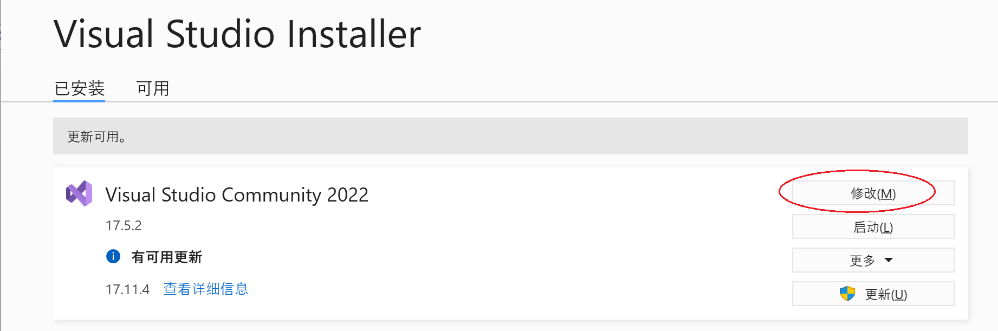
<https://visualstudio.microsoft.com/zh-hans/downloads/>

如下链接可直接下载Visual Studio Community 2022

<https://c2rsetup.officeapps.live.com/c2r/downloadVS.aspx?sku=community&channel=Release&version=VS2022&source=VSLandingPage&cid=2030:32d4696846fc4efca8838a9712caabef>

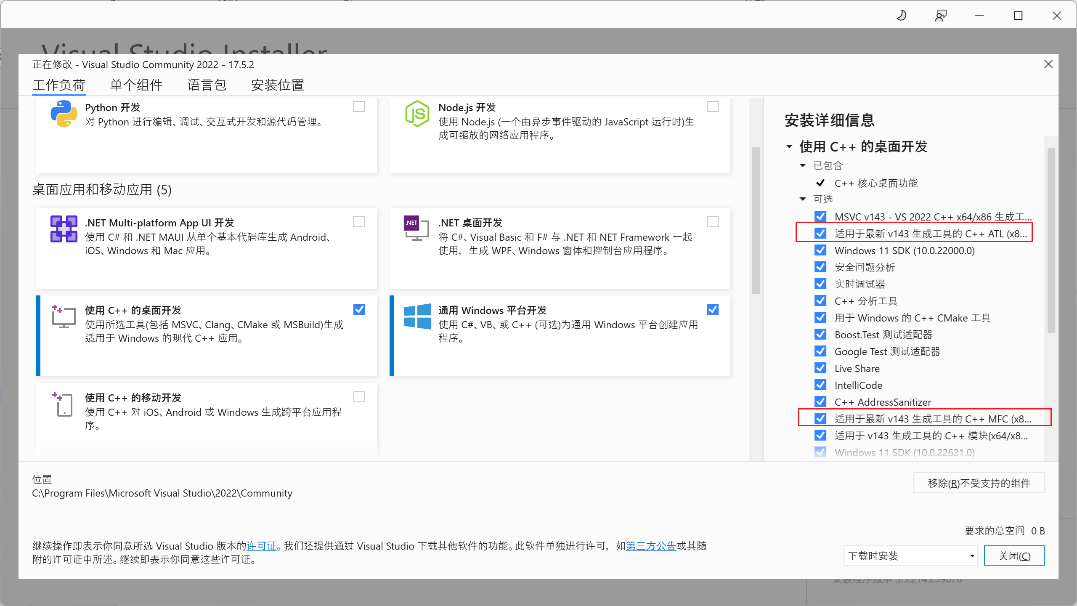
#### 安装MFC支持包

如果安装时没有选择MFC的相关支持包，可通过Visual Studio Installer修改配置。



确认已经勾选如下软件包（v143为VS 2022对应的生成工具版本号，若安装的是VS 2019则为v142）。

* C++ATL for v143生成工具(x86和x64)
* C++MFC for v143生成工具(x86和x64)



## 创建一个MFC应用程序（示例：基于对话框类型）

实验中的示例代码采用的是基于对话框的MFC应用程序。在Visual Studio下，可以使用图形化的MFC应用程序向导来辅助创建。

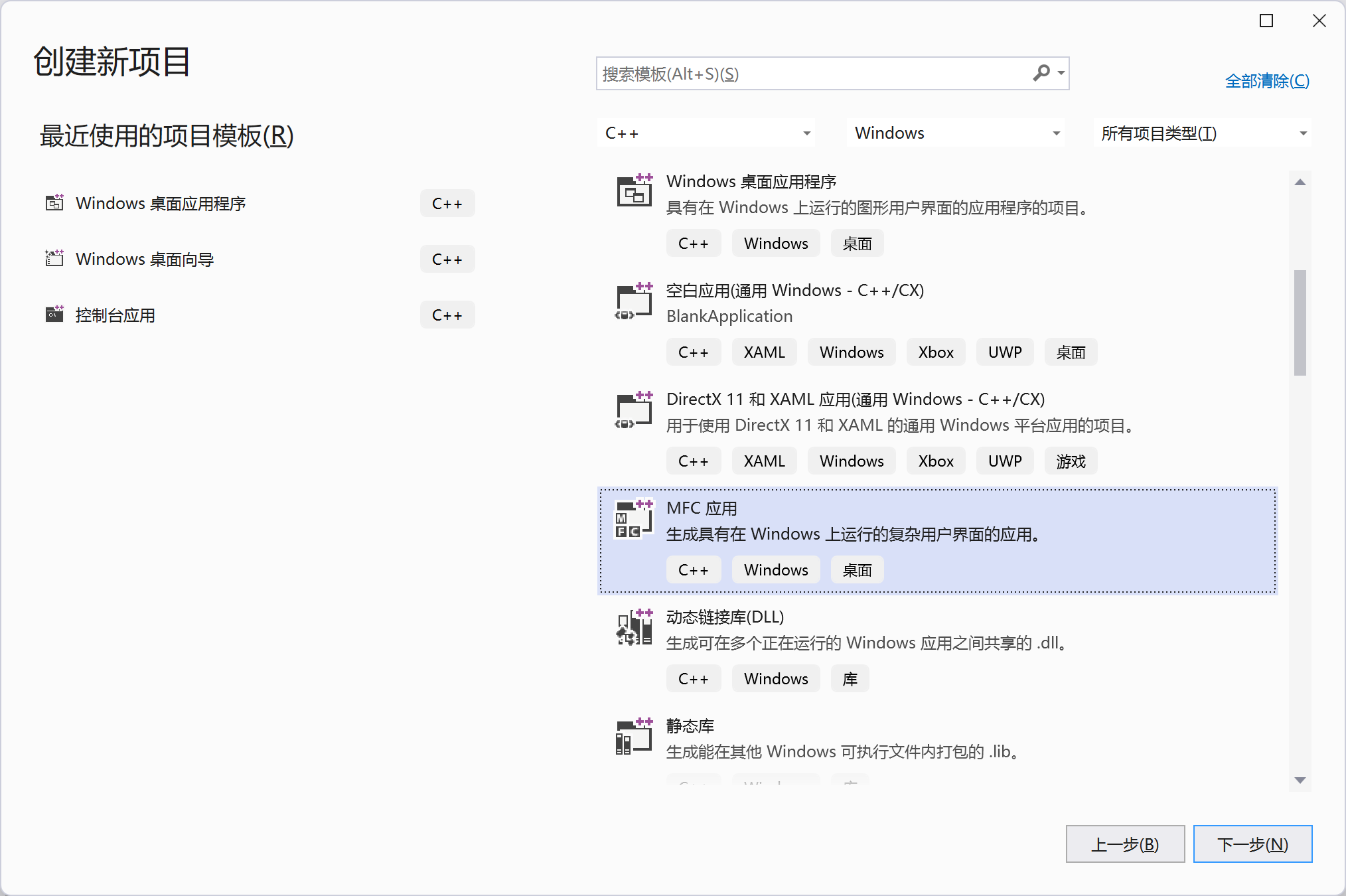
MFC 应用程序向导 | Microsoft Learn

<https://learn.microsoft.com/zh-cn/cpp/mfc/reference/mfc-application-wizard?view=msvc-170>

MFC 应用程序向导生成一个应用程序，该应用程序经过编译后可实现 Windows 可执行文件 (.exe) 应用程序的基本功能。 MFC 初学者应用程序包括 C++源 (.cpp) 文件、资源 (.rc) 文件、头文件 (.h) 文件和项目 (.vcxproj) 文件。 这些初学者文件中生成的代码基于 MFC。

例如，在Visual Studio 2022下创建一个基于对话框的MFC应用程序。

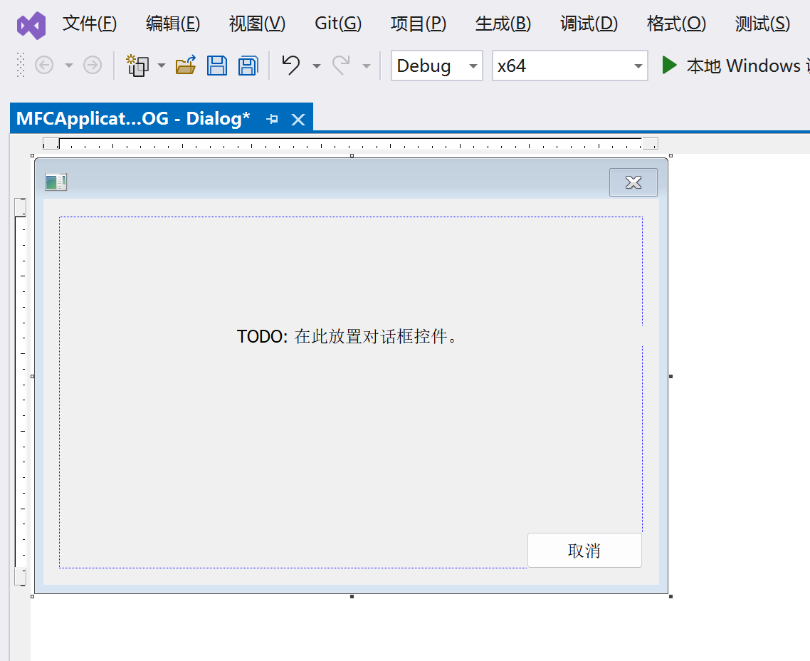
新建项目时需要选择“MFC应用”，如下，



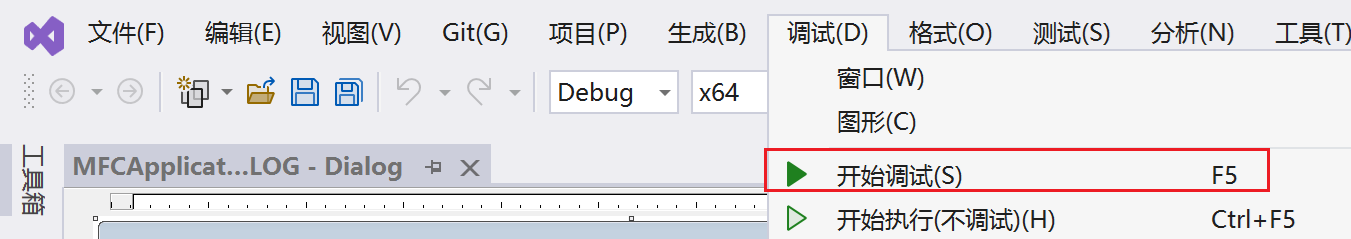
新建一个名为“MFCApplicationTest”的项目，应用程序的类型选择基于“基于对话框”，如下，



向导会自动生成项目所需的诸多文件，并设计一个基本的对话框窗口。如下，



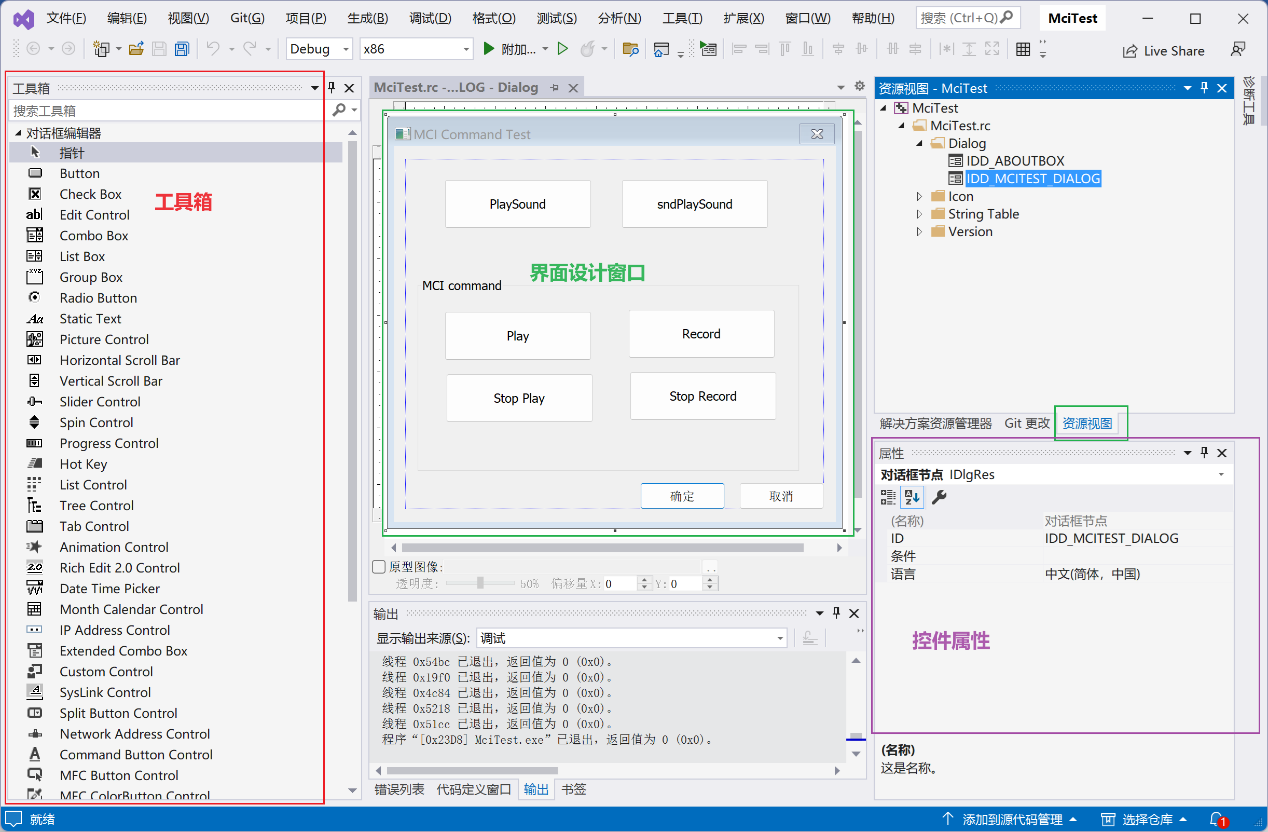
至此，向导已经生成了项目所需的各代码文件的框架，主菜单“生成”→“生成解决方案”即可生成可执行的EXE文件。随后可以以调试方式运行该程序，或直接运行（不调试），如下图。



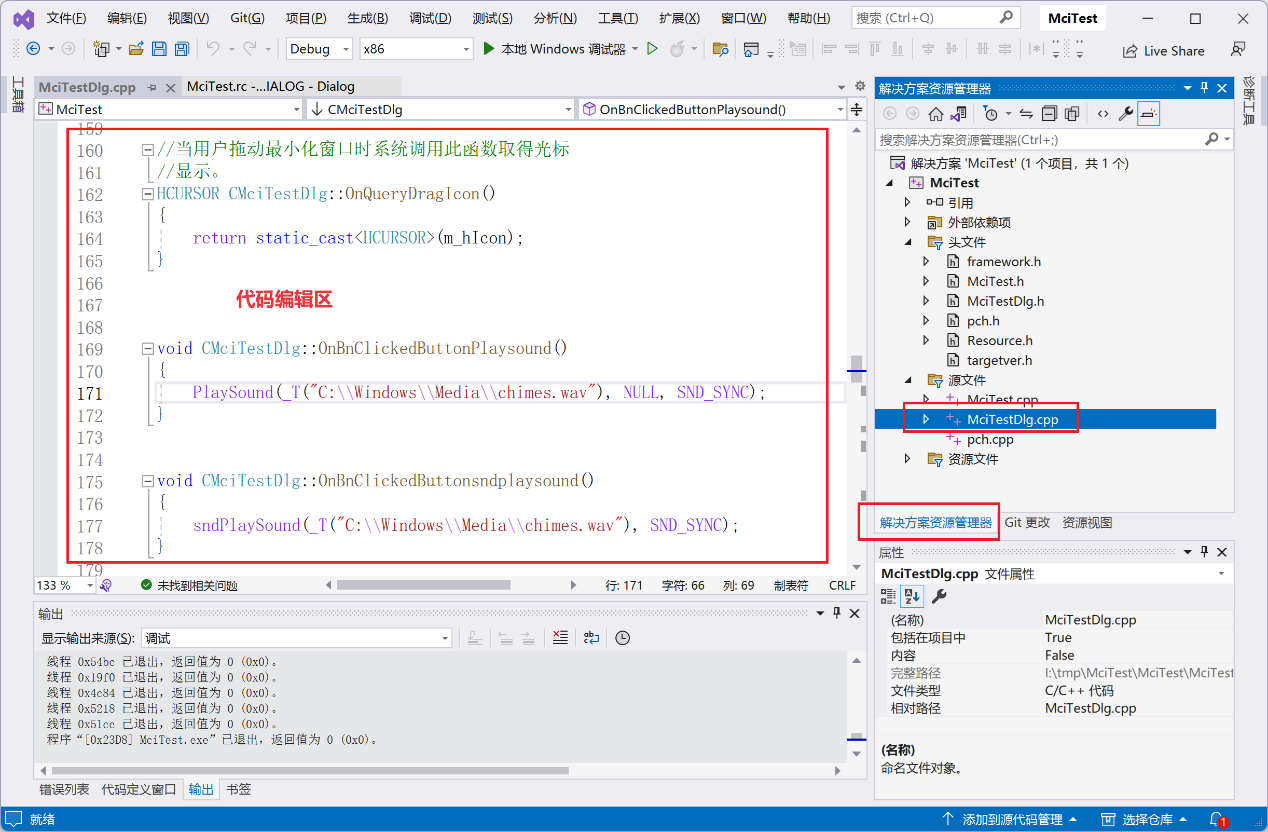
运行后即可看到向导生成的框架程序的界面如下。可以看到，这种所见即所得的程序设计方式大大提高了程序员UI设计工作的效率。



Visual Studio提供了可视化的UI设计界面，操作简洁。回顾一下上述步骤，用到了如下图所示的“界面设计窗口”、“工具箱”、“控件属性窗口”。若关闭了“界面设计窗口”，可通过图中“资源视图”中对应DIALOG调出；若“资源视图”也关闭了，主菜单“视图”→“其他窗口”→“资源视图”可调出。

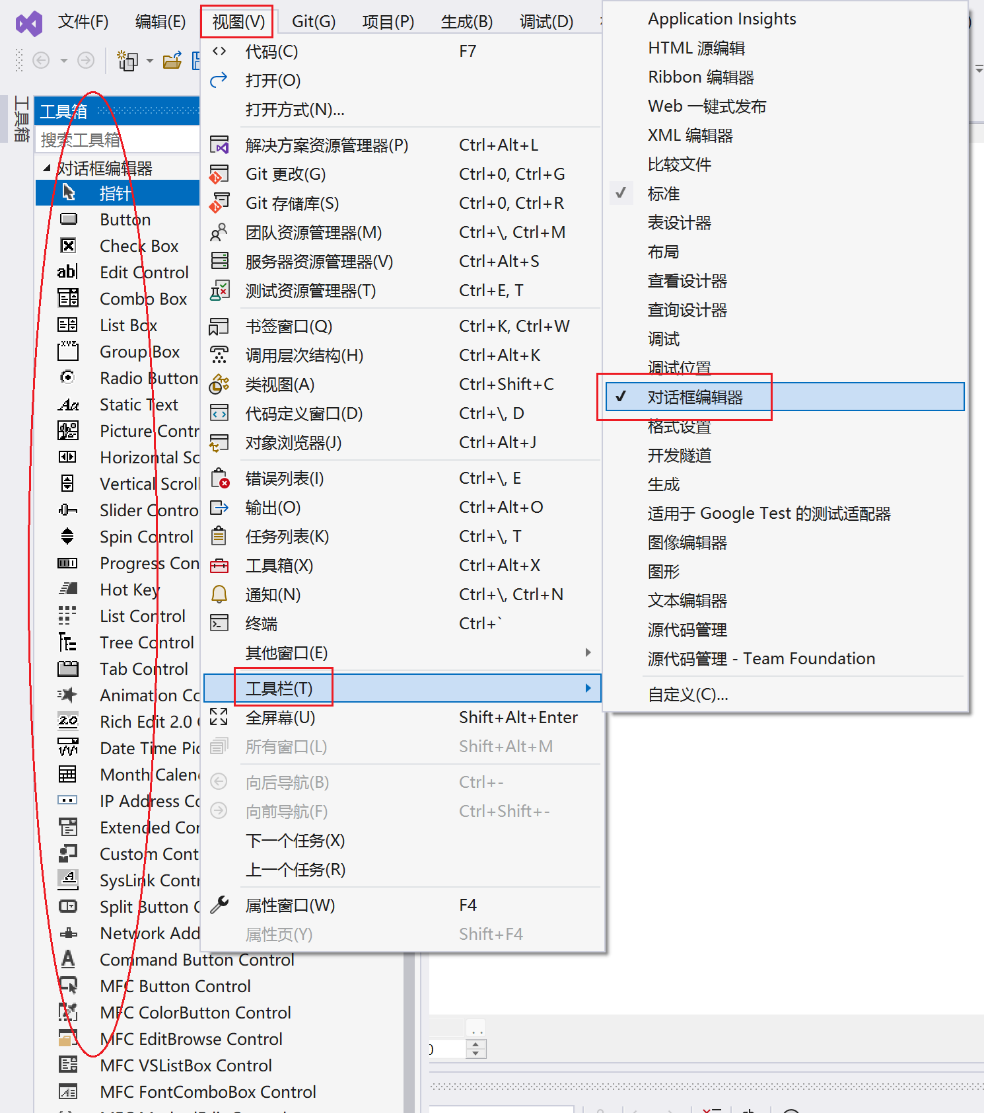


通过“解决方案资源管理器”可打开对应代码文件，若“解决方案资源管理器”被关闭，主菜单“视图”→“解决方案资源管理器”可调出。



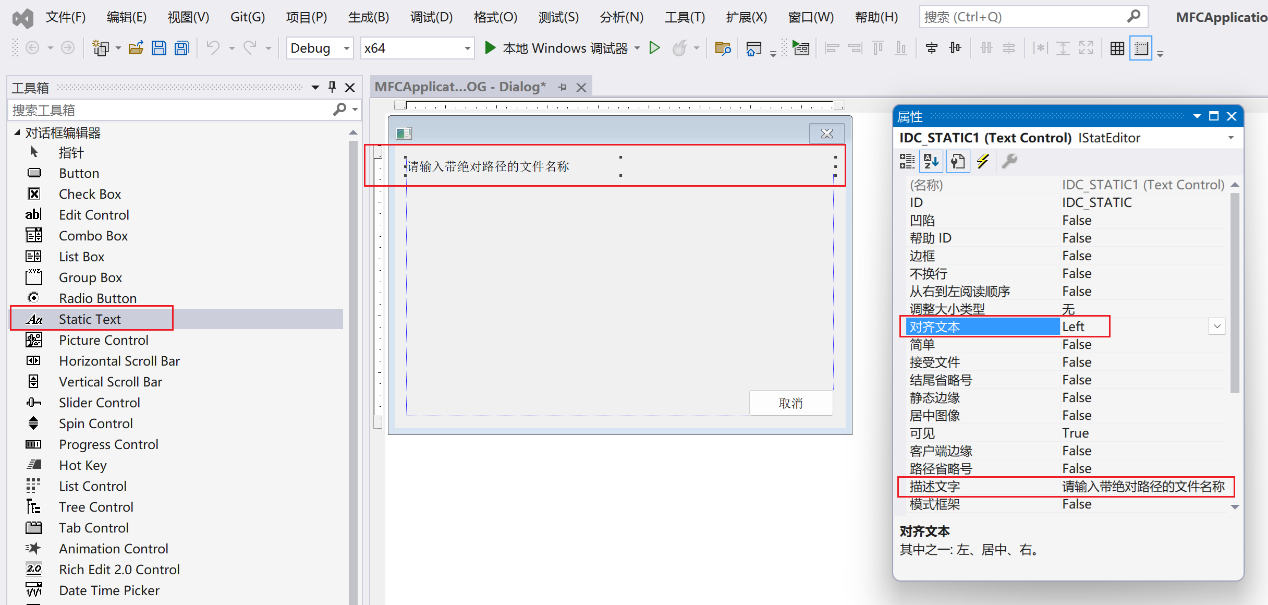
## 基于对话框的MFC应用程序的UI设计

Visual Studio提供了系列图形化控件可用于基于对话框的MFC程序的UI设计。主菜单“视图”→“工具栏”→“对话框编辑器”，如下所示。



### 常用控件：Static Text，输出提示文本

若需要在界面上放置静态提示文字，可使用“Static Text”控件，并通过图形化界面修改其属性。如下，



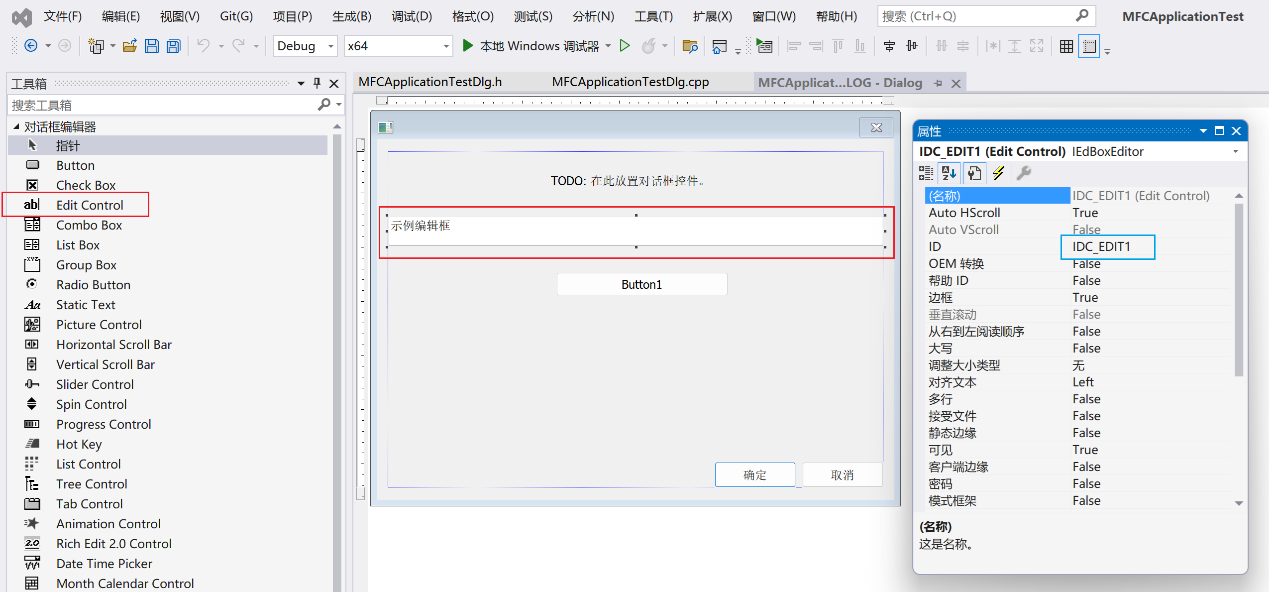
“Static Text”是对话框中最简单的控件，在UI设计阶段可以在设置属性的图形化界面修改其显示的字符串；在程序运行过程中，也可以通过代码修改所显示的字符串。例如，如下代码可以修改ID为“IDC\_STATIC”的“Static Text”显示文字。

//设置ID为IDC\_STATIC的Static Text所显示的字符串

GetDlgItem(IDC\_STATIC)->SetWindowText("Hi USTCer");

### 常用控件：Edit Control，输入字符串

若需要获取用户输入的字符串，可以使用“Edit Control”控件。通过图形化界面可以设置该控件的属性，如是否带滚动条等，如下图。



例如，如下代码是一种可能的获取Edit Control控件中用户输入字符串的方式。

char sz[20];

//获取图形界面上ID为IDC\_EDIT1的Edit Control中用户输入的字符串

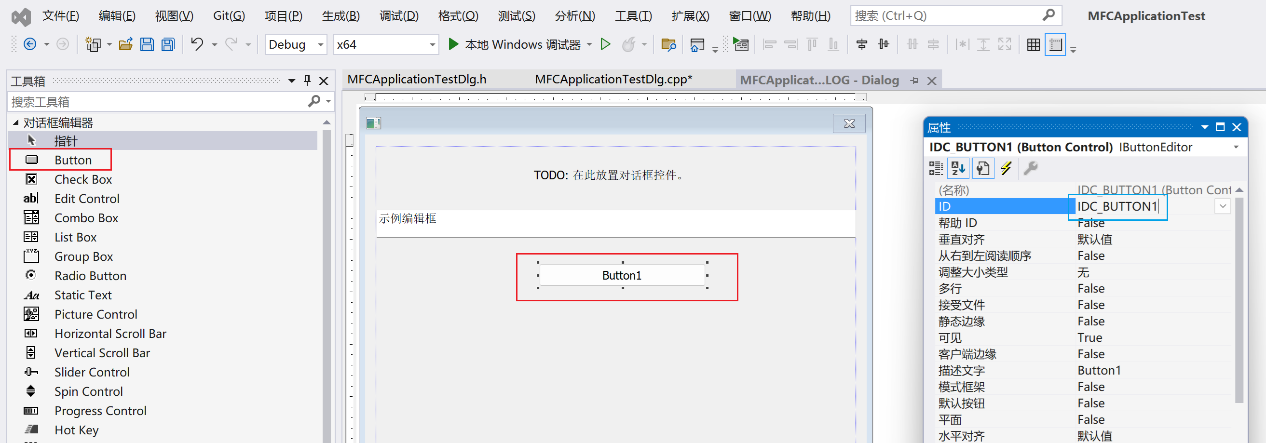
GetDlgItem(IDC\_EDIT1)->GetWindowText(sz, 20);

//弹出一个图形化的提示框，显示字符串sz

AfxMessageBox(sz);

### 常用控件：Button，接受用户动作执行一段代码

按钮是UI界面上最常用的控件。在Visual Studio的IDE中，鼠标双击界面上的按钮即自动切换到该按钮对应动作（函数）的代码编码界面。



上图所示，ID为“IDC\_BUTTON1”的按钮，双击后会自动生成一个函数名称，如下所示。在该函数中填写所需代码即可。

void CMFCApplicationTestDlg::OnBnClickedButton1()

{

//设置ID为IDC\_STATIC的Static Text所显示的字符串

GetDlgItem(IDC\_STATIC)->SetWindowText("Hi USTCer");

char sz[20];

//获取图形界面上ID为IDC\_EDIT1的Edit Control中用户输入的字符串

GetDlgItem(IDC\_EDIT1)->GetWindowText(sz, 20);

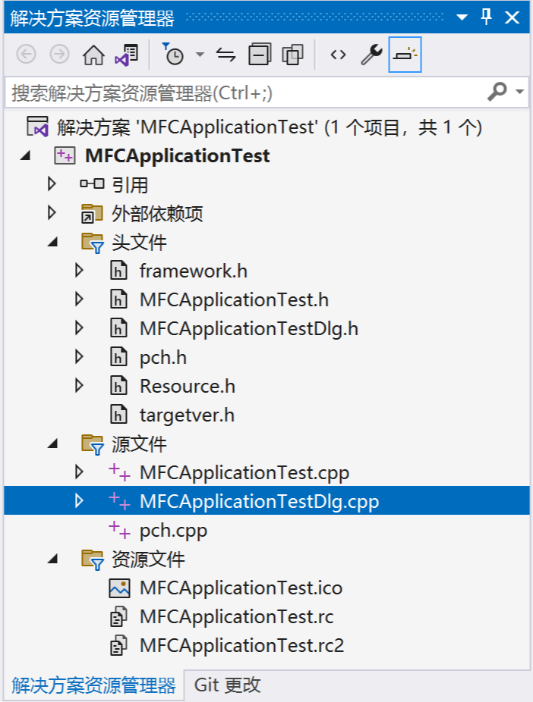
//弹出一个图形化的提示框，显示字符串sz

AfxMessageBox(sz);

}

## 进阶：了解向导生成的类

若新建项目的上述基于对话框的MFC应用程序的名称是“MFCApplicaitonTest”，则在IDE图形化界面的“解决方案资源管理器”窗口可以看到如下所示文件。



而在项目所在目录，能看到如下文件和子目录

|  |  |
| --- | --- |
| 文件/子目录名称 |  |
| framework.h |  |
| MFCApplicationTest.aps |  |
| MFCApplicationTest.cpp |  |
| MFCApplicationTest.h |  |
| MFCApplicationTest.rc | 和图形化界面有关资源的描述文件 |
| MFCApplicationTest.sln |  |
| MFCApplicationTest.vcxproj |  |
| MFCApplicationTest.vcxproj.filters |  |
| MFCApplicationTest.vcxproj.user |  |
| MFCApplicationTestDlg.cpp | 用于控制对话框界面的类 |
| MFCApplicationTestDlg.h | 用于控制对话框界面的类 |
| pch.cpp |  |
| pch.h |  |
| resource.h |  |
| targetver.h |  |
| [res] | 和图形化界面有关的资源，如程序图标 |
| [x64] | 编译、链接过程的中间文件和生成的EXE |

需要自己编写的代码主要在“MFCApplicationTestDlg.cpp”和“MFCApplicationTestDlg.h”这两个文件中。查看“MFCApplicationTestDlg.h”可以看到向导自动定义了一个类（CMFCApplicationTestDlg）来实现对话框图形界面的控制。前述按钮对应的代码是该类中的一个成员函数（OnBnClickedButton1()）。

// CMFCApplicationTestDlg 对话框

class CMFCApplicationTestDlg : public CDialogEx

{

public:

CMFCApplicationTestDlg(CWnd\* pParent = nullptr); // 标准构造函数

protected:

// 生成的消息映射函数

virtual BOOL OnInitDialog();

afx\_msg void OnSysCommand(UINT nID, LPARAM lParam);

afx\_msg void OnPaint();

afx\_msg HCURSOR OnQueryDragIcon();

DECLARE\_MESSAGE\_MAP()

public:

afx\_msg void OnBnClickedButton1();

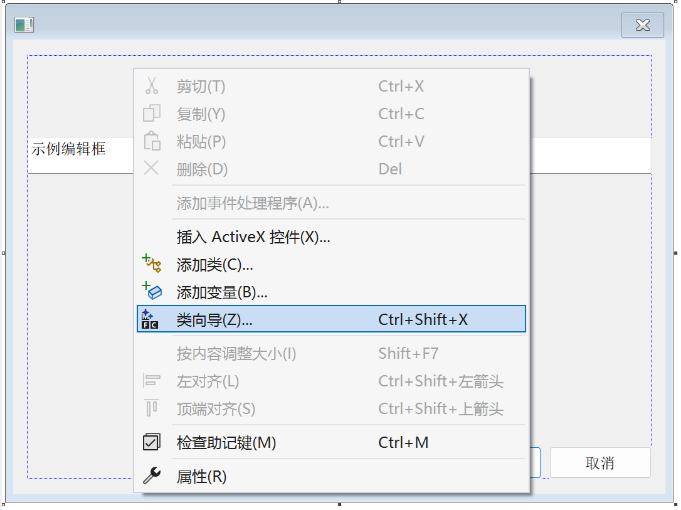
};

类CMFCApplicationTestDlg中定义的函数OnInitDialog()，是图形化界面初始化（显示前）调用的函数，一些变量的初始化可以放在此处。

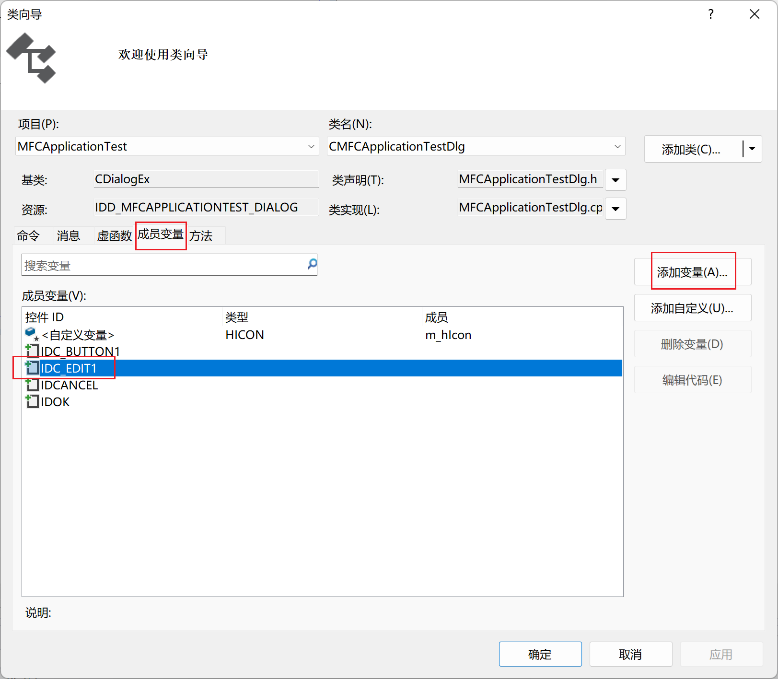
程序员可以在类CMFCApplicationTestDlg中增加新的成员变量、成员函数，从而丰富程序功能。

## 进阶：添加类变量实现UI控件交互

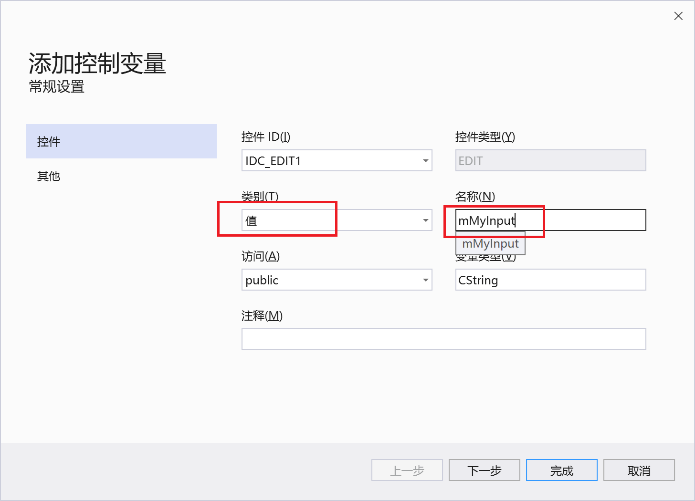
可以通过类向导（设计界面下，右键）在类CMFCApplicationTestDlg中增加和UI控件（如Edit Control）绑定的成员变量。



给界面上的“Edit Control”添加一个对应的变量。



选择变量的类型为“值”，假设变量的名称是“mMyInput”。



查看“MFCApplicationTestDlg.h”可以看到向导自动在类CMFCApplicationTestDlg中添加了一个成员变量mMyInput，代码如下。

CString mMyInput;

在对话框窗口上添加一个“Button”控件，修改其ID为“IDC\_BUTTON\_GetInput”。在其对应代码中采用刚才定义的变量mMyInput来实现交互，如下。

void CMFCApplicationTestDlg::OnBnClickedButtonGetinput()

{

//UpdateData(FALSE)表示将变量中的数据更新到对应的界面控件中

//UpdateData(TRUE)表示将界面上的数据更新到与之对应的变量中

UpdateData(TRUE);

AfxMessageBox(mMyInput);

}

更多关于类向导使用技巧可参考如下博文

类向导简介（例子：简单计算器）\_mfc类向导的成员变量

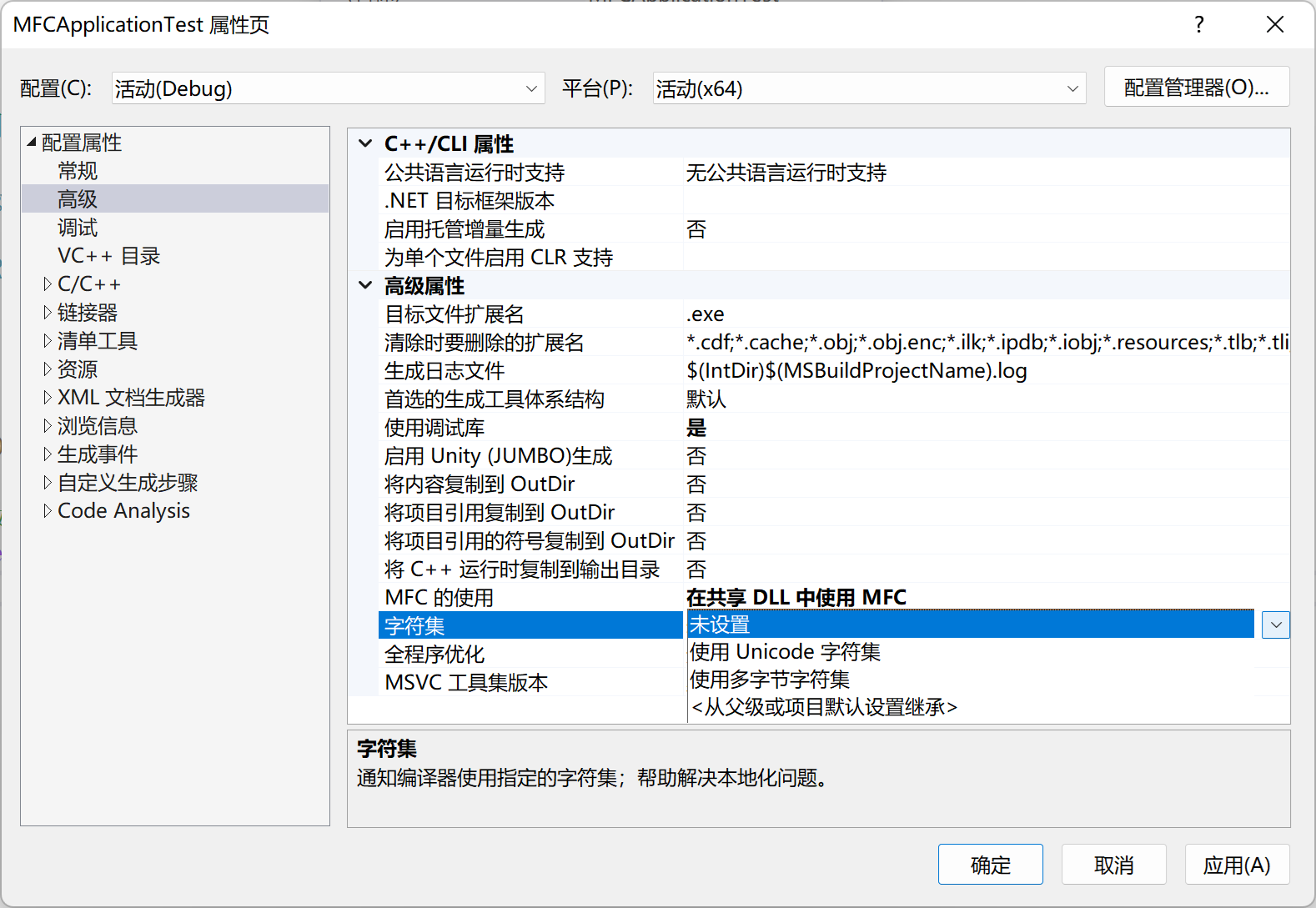
<https://blog.csdn.net/qq_38834877/article/details/103172214>

【MFC实践】基于MFC向导C++制作计算器（附文件）\_c++ mfc

<https://blog.csdn.net/m0_73589720/article/details/135406115>

## 进阶：编译、链接选项的配置

主菜单“项目”→“属性”，可打开项目属性的配置界面。例如，配置属性的“高级”页，可以设置所使用的字符集，如下图。**建议初学者将该项设置为“未设置”**，而不要采用“使用Unicode字符集”（使用Unicode字符集时函数参数为字符串时，要先转换为Unicode格式，很多细节较易出错[[3]](#footnote-3)）。



配置属性的“VC++目录”页可以设置头文件路径等信息，“C/C++”页可以设置编译选项等信息，“链接器”页可以指定需要引用的静态库（\*.lib）等。初学者在学习过程中可以浏览配置属性各个配置页。

## VC5/VC6项目移植至高版本Visual Studio

部分早期的实验示例代码是在VisualC++5/6的IDE下编写的，这些示例代码的项目（\*.dsw）在转换至高版本的Visual Studio项目（\*.sln）的时候，需要安装一些组件才可以顺利移植。

### Visual Studio 2019移植记录

* 在Visual Studio 2019下，读取VC6生成的项目转换后，Build项目的错误信息

MSB8041：此项目需要 MFC 库

解决方法：

打开Visual Studio Installer → 修改 → 单个组件，选择安装：

C++ATL for v142生成工具(x86和x64)

C++MFC for v142生成工具(x86和x64)

C++ v14.21 ATL for v142生成工具(x86和x64)

C++v14.21 MFC for v142生成工具(x86和x64)

如果还是出现同样错误，安装

适用于最新v142生成工具的C++ATL

适用于最新v142生成工具的C++ MFC

原文链接：<https://blog.csdn.net/weixin_45814496/article/details/120777219>



* 解决“错误 D8016 “/ZI”和“/Gy-”命令行选项不兼容 ”问题\_cl : 命令行 error d8016: “/zi”和“/gy-”命令行选项不兼容

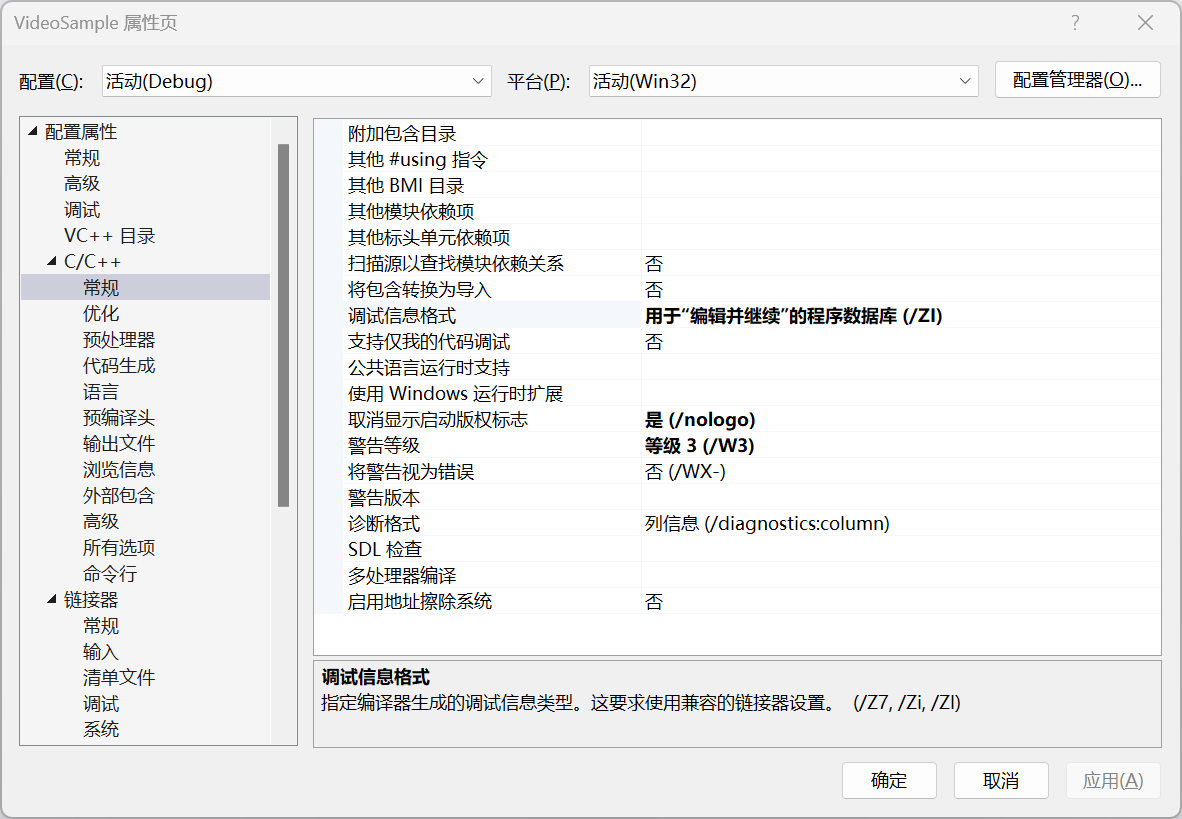
<https://blog.csdn.net/lyj_viviani/article/details/51487877>

Build出现错误

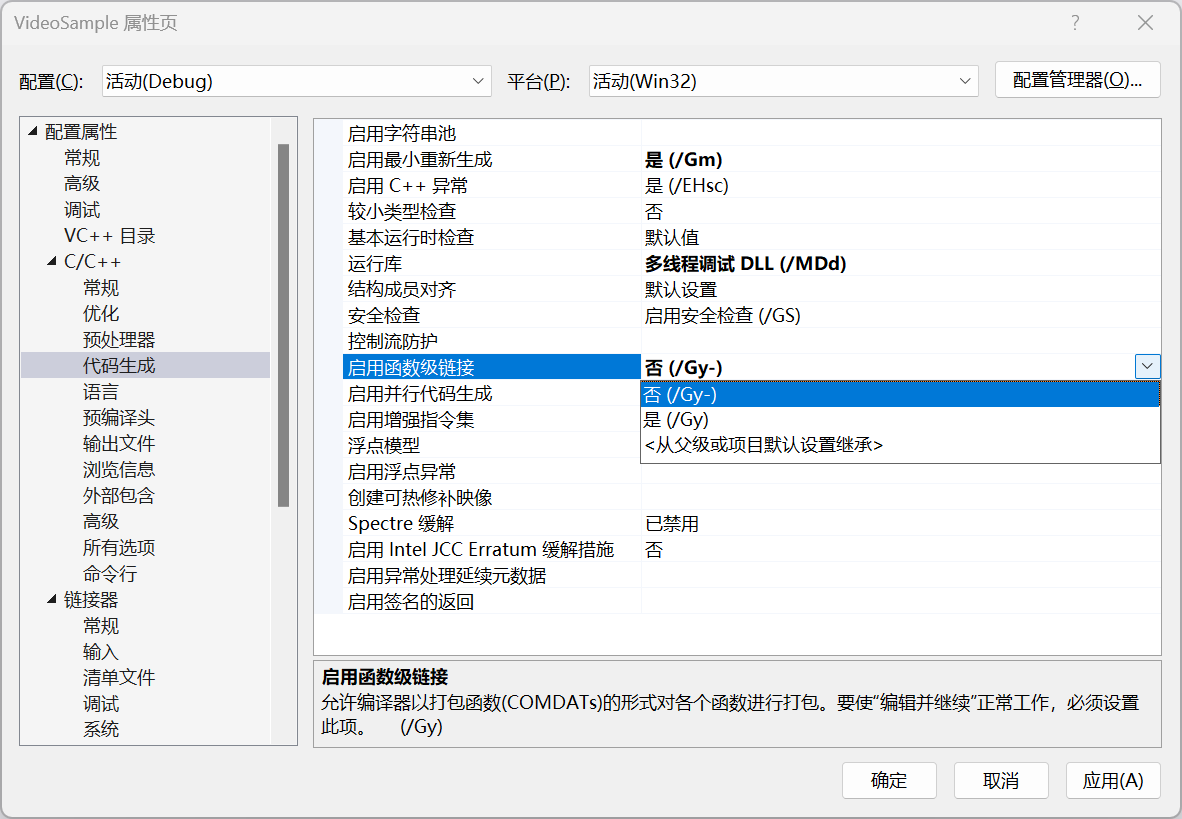
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 严重性 | 代码 | 说明 | 项目 | 文件 |
| 错误 | D8016 | “/ZI”和“/Gy-”命令行选项不兼容 | VideoSample | C:\Users\fama\Downloads\VideoSample\_CallBack\_show20051120\cl |

我们需要手动改变/ZI命令行选项 或者 /Gy命令行选项进入“项目”→“属性”→>“C/C++”

（1）→“常规”→“调试信息格式”→选择“程序数据库(/Zi)”或“无”



（2）→“代码生成”→“启用函数集链接” →选择“是 (/Gy)”



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 严重性 | 代码 | 说明 | 项目 | 文件 |
| 错误 | C2440 | “static\_cast”: 无法从“void (\_\_thiscall CVideoSampleDlg::\* )(WPARAM,LPARAM)”转换为“LRESULT (\_\_thiscall CWnd::\* )(WPARAM,LPARAM)” | VideoSample | C:\Users\fama\Downloads\VideoSample\_CallBack\_show20051120\VideoSampleDlg.cpp |

<https://www.debugease.com/vc/1950801.html>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 严重性 | 代码 | 说明 | 项目 | 文件 |
| 错误 | LNK2026 | 模块对于 SAFESEH 映像是不安全的。 | VideoSample | C:\Users\fama\Downloads\VideoSample\_CallBack\_show20051120\VFW32.LIB(MSVFW32.dll) |

常见报错：error LNK2026: 模块对于 SAFESEH 映像是不安全的

解决方法：右键打开项目属性 → 链接器 → 命令行 → 其他选项 (D) 中加入 /SAFESEH:NO 这句，点击应用即可。

### Visual Studio 2022移植记录

Visual Studio 2022 移植、迁移和升级项目 | Microsoft Learn

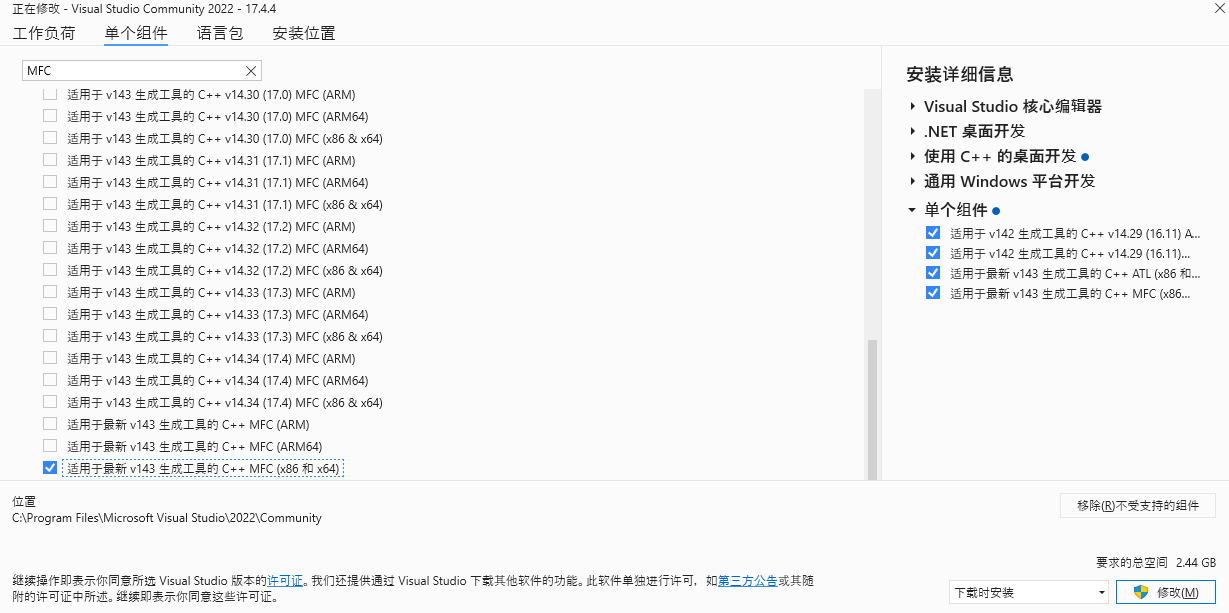
<https://learn.microsoft.com/zh-cn/visualstudio/releases/2022/port-migrate-and-upgrade-visual-studio-projects>

我们尝试保留与早期版本的后向兼容性，如 Visual Studio 2019、Visual Studio 2017、Visual Studio 2015、Visual Studio 2013 和 Visual Studio 2012。 但是，对某些项目类型的支持会随着时间的推移而更改。 较新版本的 Visual Studio 可能根本不支持某些项目，或者可能需要更新项目，使其不再向后兼容。

Visual Studio 2022 中已经不再支持打开.dsw项目，所以需要先用Visual Studio 2019 打开项目，由Visual Studio 2019完成转换。

打开Visual Studio 2019转换后的sln，Build项目的错误信息：

error MSB8041: 此项目需要 MFC 库。从 Visual Studio 安装程序(单个组件选项卡)为正在使用的任何工具集和体系结构安装它们。



# 附录：Visual C++下的多媒体开发

## Visual C++多媒体开发方法

### 使用OLE技术

OLE是一种动态信息交换协议，它通过一个由OLE包容器（Container）和OLE服务器（Server）组成的系统来运作。OLE技术允许一个应用程序调用另一个应用程序来提供服务。在多媒体开发中，一个多媒体应用程序要求的服务可能是播放音频，也可能是更复杂的音频和视频同时播放。实现OLE的方法，一种是直接连接或嵌入；另一种使用包。如下是几个调用系统缺省应用程序的示例：

ShellExecute(NULL,"open","c:\\windows\\genneral.txt",

NULL,NULL,SW\_SHOWNORMAL);

//用Notepad.exe打开一个文本文件

ShellExecute(NULL,"open","c:\\ windows [\\media\\ding.wav"](file:///\\media\\ding.wav%22),

NULL,NULL,SW\_SHOWNORMAL);

//用媒体播放器播放一个WAVE文件

ShellExecute(NULL,"open","http://202.38.75.33",

NULL,NULL,SW\_SHOWNORMAL);

//用缺省的浏览器打开网页

### 运用OCX

OCX（OLE Control eXtensions）可以分为两类，一类是OLE公用控件；另一类是OLE自定义控件。在VC中，提供了用于提高代码重用的重要工具—组件平台（Compenent Gallery），提供了大量的公用OCX，其中MCI32.ocx就是用于管理Media Control Interface(MCI，媒体控制接口)设备的多媒体文件的录音和播放。该控件可以显示一套用于将MCI命令传向设备的推压式按钮，这些设备包括音频板、MIDI序列发生器、CD-ROM驱动器、音频CD播放器、视频光盘播放器以及视频磁带录音机和播放器。该控件还支持Video for Windows AVI文件的播放。

### 创建和使用DLL

DLL是一个包含了若干函数的可执行模块，可以实现应用程序的共享代码和资源。在VC下可以考虑利用DLL来扩充多媒体系统的功能。例如在DLL中定义诸如播放动画、声音等的函数，然后在主程序中通过LoadLibrary函数装载DLL，即可以调用DLL中的函数来实现所需要的功能。可以在有关的站点找到一些第三方厂家的具有多媒体特性的DLL，如AAPLAY.DLL可将3DS的动画文件进行播放。

### 编制MFC类

VC下，可以使用其他公司开发的多媒体类，从而可以利用现有类的功能实现多媒体应用程序。例如，CtegMM类提供对多媒体设备的控制，可以播放CD音频、WAVE文件、MIDI文件等。也可以定义自己的类，更自由的实现有关多媒体功能。

### Windows类MCIWnd的应用

MCIWnd（Meida Control Interface Windows）是一个预包装的MCI播放器的Windows类（不是一个CV类），在Video for Windows（VFW）中实现，开发者几乎不用编程就可以使用它提供的各种媒体控制如暂停、速度、大小和音量等。VC下使用MCIWnd类，必须包括头文件VFW.H，连接设置中指出VFW32.LIB，然后用MCIWndCreat函数来创建MCIWnd。

### 使用Windows多媒体API函数

Windows多媒体系统函数在DLL中，在开发Windows多媒体应用程序时，最简单的方法是利用Windows的媒体控制接口（MCI）来实现。媒体控制接口属高层音频服务，其MCI设备驱动程序封装了操作波形设备的许多细节，因而编程量小，简单易用。（在VC5中，编制自己的多媒体程序时需要将winmm.lib库连接。有时还需要包括头文件mmsystem.h）。

利用MCI高级函数编制的应用程序不能进行录入和播出数据的实时处理。如果需要做较高要求的数据实时处理，则需要利用Windows的多媒体开发工具（MDK）所提供的与设备无关低层音频服务接口函数。

## Windows多媒体API函数

### Windows多媒体API函数

Windows提供低级和高级两组多媒体系统函数，常用函数接口如下表所示。有关函数的接口请参阅VC的联机帮助。

|  |  |  |
| --- | --- | --- |
| MCI接口 | 解释 | 级别 |
| PlaySound  SndPlaySound | 是两个发声函数，只执行单一的播放WAVE文件功能的高级函数 | 处于多媒体接口的顶点 |
| (命令-消息接口)  mciSendCommand  mciGetDeviceID | 两个高级命令接口⎯命令-消息和命令-字符串执行相同的功能，区别是单词和数字的区别⎯最后的执行结果是一样的 | 高级MCI接口 |
| (命令-字符串接口)  mciSendString |
| (命令-字符串和命令-消息接口)  mciGetErrorString  mciSetYieldProc |
| (常用的多媒体文件I/O函数)  mmioOpen mmioClose mmioRead  mmioWrite mmioSendMessage  mmioDescend mmioAscend | 高级MCI函数即封装调用了这些I/O函数，用于处理RIFF文件，可执行各种复杂的操作 | 低级多媒体函数 |
| TimeGetDevCaps timeBeginPeriod  TimeSetEvent timeKillEvent timeEndPeriod | 实现媒体播放等功能的高精度计时 | 更低层的计时器函数 |

### WAVE格式音频的相关函数

在Windows多媒体API函数中，由一部分专门用于控制WAVE格式的音频数据，下表将列出在实验样本程序中使用到的API函数，其他相关的函数请参阅VC的联机帮助。

|  |  |
| --- | --- |
| 函数名称 | 函数功能 |
| PlaySound | 播放WAVE文件 |
| SndPlaySound | 播放WAVE文件 |
| WaveInOpen | 打开WAVE格式录音设备 |
| WaveInClose | 关闭WAVE格式录音设备 |
| WaveInGetNumDevs | 检查系统中录音设备的数目 |
| WaveInAddBuffer | 将一个录音缓冲区送给录音设备 |
| WaveInPrepareHeader | 准备一个缓冲区用于录音 |
| WaveInStart | 开始录音 |
| WaveInStop | 停止录音 |
| WaveOutOpen | 打开WAVE格式放音设备 |
| WaveOutClose | 关闭WAVE格式放音设备 |
| WaveOutGetNumDevs | 检查系统中放音设备的数目 |
| WaveOutPause | 暂停放音 |
| WaveOutPrepareHeader | 准备一个缓冲区用于播放 |
| WaveOutSetVolume | 设置放音的音量 |

### API使用示例

#### sndPlaySound调用格式：

BOOL sndPlaySound( LPCSTR lpszSound, UINT fuSound);

入口参数：

lpszSound 要播放的WAVE文件的名称

fuSound 播放参数，函数的返回值和该参数有关。

以下为使用示例：

*//播放系统声音“ding.wav”*

sndPlaySound("ding.wav",SND\_ASYNC);

*//停止声音的播放*

sndPlaySound("ding.wav",NULL);

#### PlaySound调用格式：

BOOL PlaySound( LPCSTR pszSound,

HMODULE hmod, DWORD fdwSound);

入口参数：

pzSound 要播放的WAVE文件的名称

hmod 指定播放器句柄，fdwSound =SND\_RESOURCE时有效

fdwSound 播放参数，函数的返回值和该参数有关。

以下为使用示例：

*//使用系统缺省的播放器播放*

PlaySound("chord.wav",NULL,SND\_SYNC );

*//使用播放器“MyPlaer.exe”播放*

PlaySound("MyPlaer.exe",hmod , SND\_RESOURCE );

PlaySound("chord.wav",hmod,SND\_SYNC );

*//停止声音的播放*

PlaySound(NULL,hmod,SND\_SYNC );

#### mciSendCommand调用格式：

MCIERROR mciSendCommand(MCIDEVICEID IDDevice,

UINT uMsg,

DWORD fdwCommand,

DWORD dwParam

);

入口参数：

IDDevice 设备的ID

Umsg 用户发出的MCI命令

FdwCommand 有关Umsg的参数

DwParam 包含命令Umsg信息的结构指针

以下为使用示例：

*//声明有关的变量*

HWND hwnd;

MCIDEVICEID wDeviceID;

MCI\_OPEN\_PARMS mciopen, mciplay;

hwnd=GetActiveWindow()->m\_hWnd;

mciopen.lpstrElementName = "e:\\ding.wav";

mciopen.lpstrDeviceType = "waveaudio";

*//打开对应“waveaudio”的放音设备*

mciSendCommand(0,MCI\_OPEN,

MCI\_OPEN\_TYPE|MCI\_OPEN\_ELEMENT,

(DWORD)(LPVOID)&mciopen);

*//播放打开的文件*

wDeviceID = mciopen.wDeviceID;

mciplay.dwCallback = (DWORD)hwnd;

mciSendCommand(wDeviceID,MCI\_PLAY,MCI\_NOTIFY,

(DWORD)(LPVOID)&mciplay);

#### WaveOutOpen和WaveInOpen调用格式

MMRESULT waveOutOpen( LPHWAVEOUT phwo,

UINT uDeviceID,

LPWAVEFORMATEX pwfx,

DWORD dwCallback,

DWORD dwCallbackInstance,

DWORD fdwOpen

);

入口参数：

phwo 输出设备的句柄

uDeviceID 输出设备的ID

pwfx 波形数据格式说明

dwCallback 回调窗口的句柄

dwCallbackInstance 用户定义的句柄，不用

fdwOpen 打开设备的类型

以下为使用示例：

*//有关的变量声明*

PCMWAVEFORMAT PCMWaveFmtRecord; *//存放波形数据*

WAVEHDR WaveHeader; *//存放数据格式说明*

HWAVEOUT hWaveOut; *//输出设备句柄*

*//打开一个音频输出设备*

waveOutOpen(&hWaveOut,

WAVE\_MAPPER,

(WAVEFORMATEX\*)&PCMWaveFmtRecord,

0l,0l,0l);

*//读取数据格式信息*

waveOutPrepareHeader(hWaveOut,&WaveHeader,sizeof(WaveHeader));

*//将波形数据写入打开的输出设备*

waveOutWrite(hWaveOut,&WaveHeader,sizeof(WaveHeader));

do{}while(!(WaveHeader.dwFlags & WHDR\_DONE));

*//释放存放数据格式信息的内存*

waveOutUnprepareHeader(hWaveOut,&WaveHeader,sizeof(WaveHeader));

WaveHeader.dwFlags = 0l;

*//关闭所打开的输出音频设备*

waveOutClose(hWaveOut);

## Windows中有关多媒体的结构定义

### 结构体PCMWAVEFORMAT

PCMWAVEFORMAT结构用于描述PCM结构波形数据的格式，其结构定义如下所示：

typedef struct {

WAVEFORMAT wf;

WORD wBitsPerSample;

} PCMWAVEFORMAT;

其中WBitsPerSample是采样率

其中wf为WAVEFORMAT结构，用于描述数据的一般结构信息，它包括了所有波形文件的通用格式信息。结构定义如下所示：

typedef struct {

WORD wFormatTag;

WORD nChannels;

DWORD nSamplesPerSec;

DWORD nAvgBytesPerSec;

WORD nBlockAlign;

WORD wBitsPerSample;

WORD cbSize;

} WAVEFORMATEX;

具体参数解释如下：

wFormatTag

波形数据的格式，定义在MMREG.H文件中

nChannels

波形数据的通道数：单声道或立体声

nSamplesPerSec

采样率，对于PCM格式的波形数据，采样率有8.0 kHz, 11.025 kHz, 22.05 kHz, and 44.1 kHz等几种

nAvgBytesPerSec

数据率，对于PCM格式的波形数据，数据率等于采样率乘以每样点字节数

nBlockAlign

每个样点的字节数

wBitsPerSample

采样精度，对于PCM格式的波形数据，采样精度为8或16

cbSize

附加格式信息的数据块大小

### 设备头结构WAVEHDR

结构WAVEHDR定义了指向波形数据缓冲区的设备头，具体定义如下：

typedef struct {

LPSTR lpData;

DWORD dwBufferLength;

DWORD dwBytesRecorded;

DWORD dwUser;

DWORD dwFlags;

DWORD dwLoops;

struct wavehdr\_tag \* lpNext;

DWORD reserved;

} WAVEHDR;

lpData 波形数据的缓冲区地址

dwBufferLength 波形数据的缓冲区地址的长度

dwBytesRecorded 当设备用于录音时，标志已经录入的数据长度

dwUser 用户数据

dwFlags 波形数据的缓冲区的属性

dwLoops 播放循环的次数，仅用于播放控制中

lpNext和reserved均为保留值

注意：参考VC中mmsystem.h文件中的有关定义可以知道，PWAVEHDR, NPWAVEHDR, LPWAVEHDR,LPHWAVEIN 均 定义为指向结构WAVEHDR的指针。在实验的样本程序中，用到下列有关多媒体的结构：“HWAVEIN”、“ LPWAVEHDR”、“ MMRESULT”。

# 附录：Android Studio程序开发极简入门

2013年5月16日，在I/O大会上，谷歌推出Android Studio。Android Studio 是用于开发 Android 应用的官方集成开发环境 (IDE)。Android Studio 基于 IntelliJ IDEA 的代码编辑器和开发者工具，还提供更多可提高 Android 应用构建效率的功能。

## 安装Android Studio相关软件

### 安装 Android Studio

安装 Android Studio  |  Android Developers (google.cn)

<https://developer.android.google.cn/studio/install?hl=zh-cn>

默认的安装选项，安装结束后启动Android Studio。可以看到启动画面（Splash Screen）。



#### 关于JDK

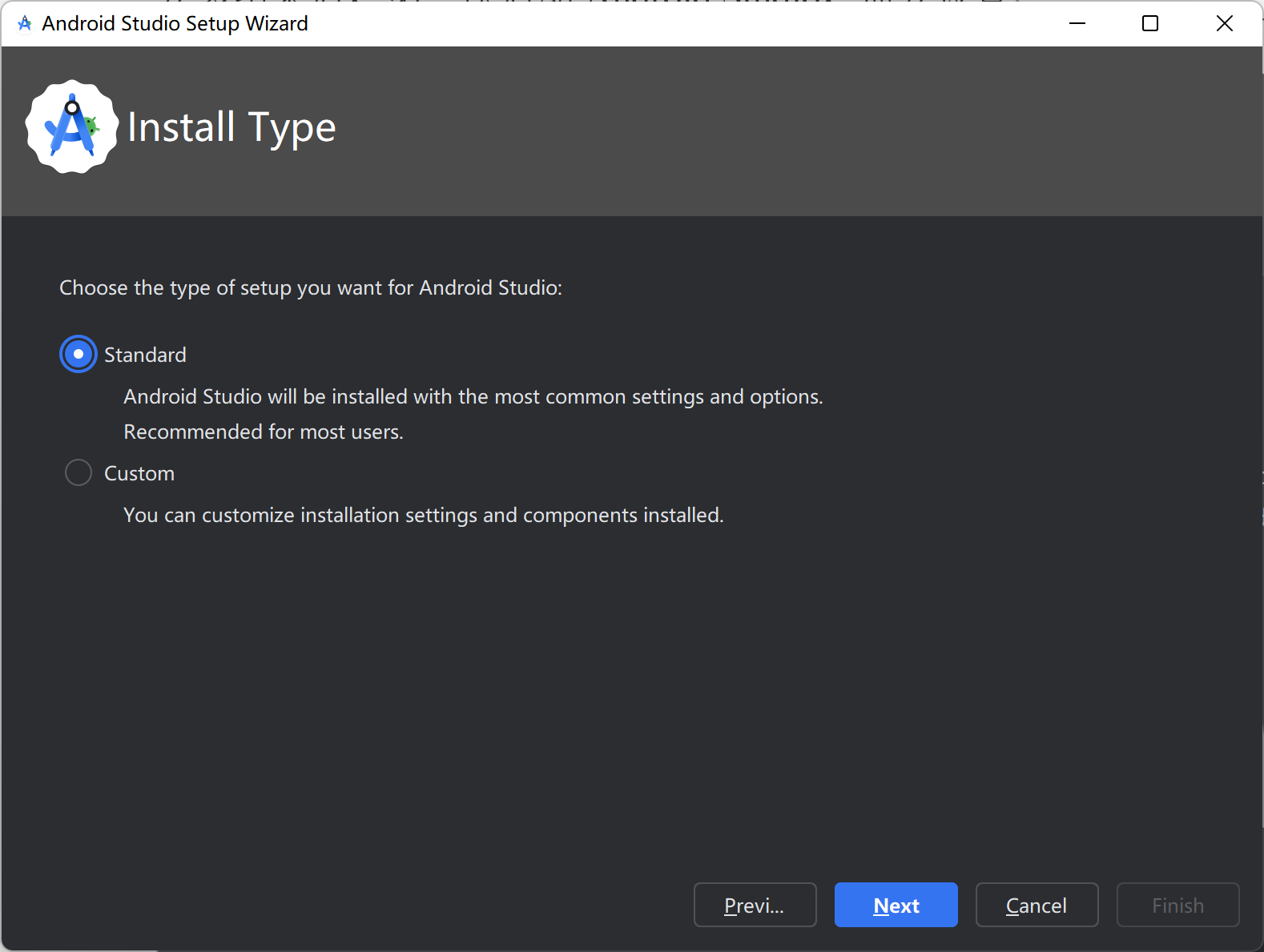
早期版本的Android Studio，需要单独安装JDK (java development kit) 即java开发工具包，很多博客的Android Studio安装教程中都有描述如何配置JDK。而较高版本的Android Studio安装包已经内置了JDK，对于初学者而言，可以不用自己单独安装JDK。但是如果需要编译早期写的程序，可能要求JDK版本要低于某个特定版本，此时就需要手动安装JDK。

### 安装Android SDK

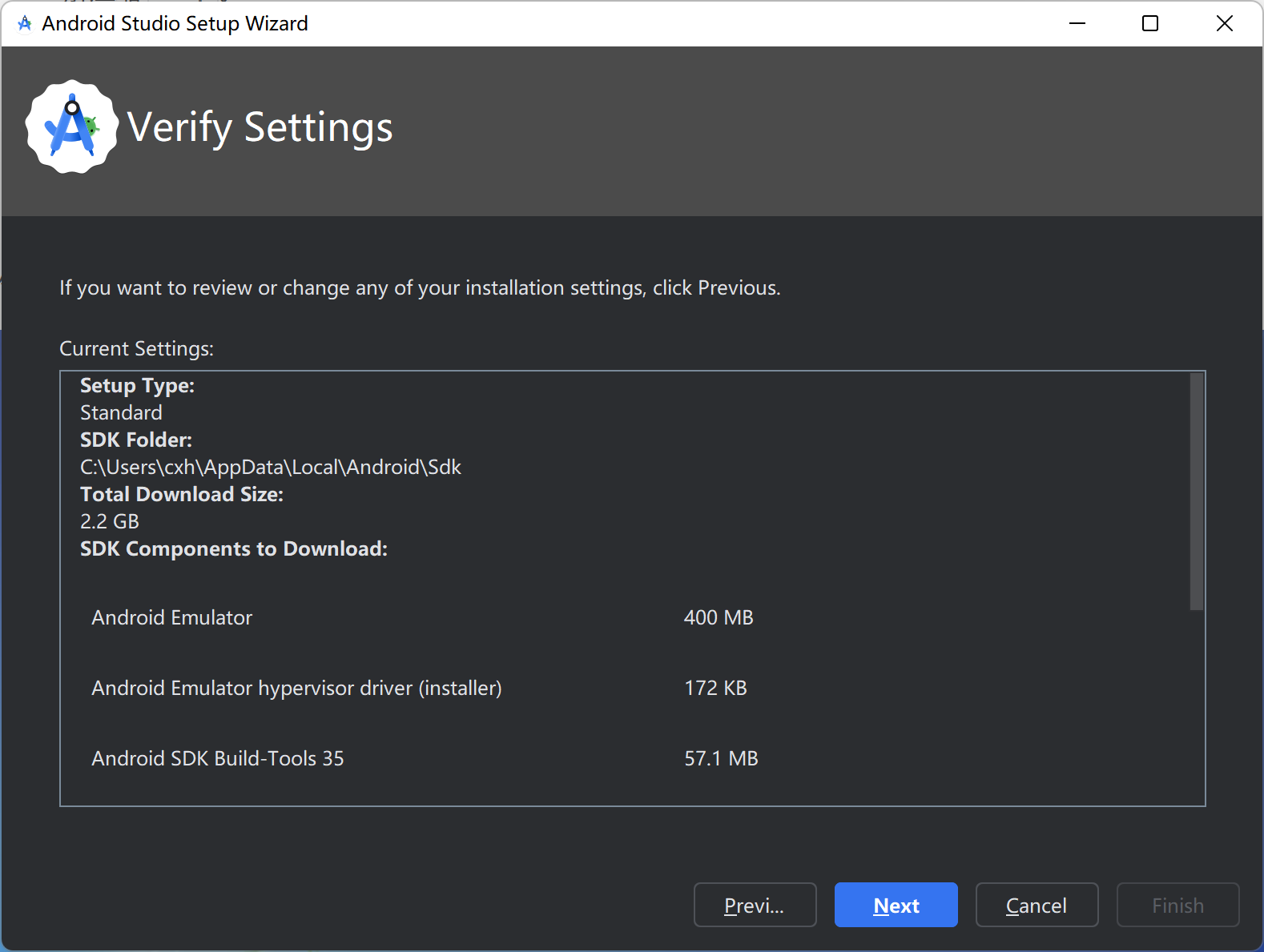
Android Studio安装结束后，第一次启动Android Studio，需要根据向导设置（安装）Android SDK。



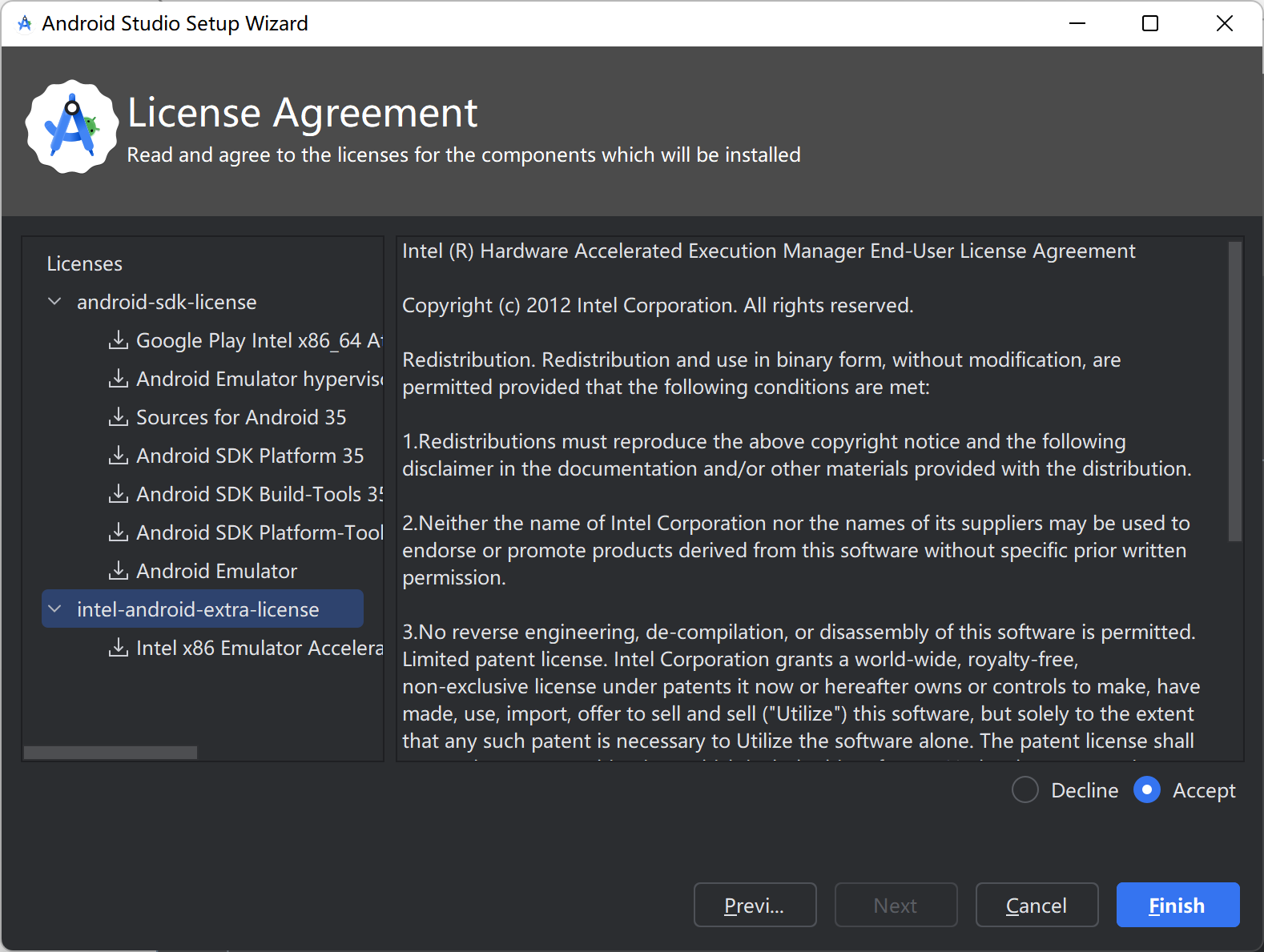
根据向导所示步骤，选择标准安装方式。



确认需要安装的具体软件包。



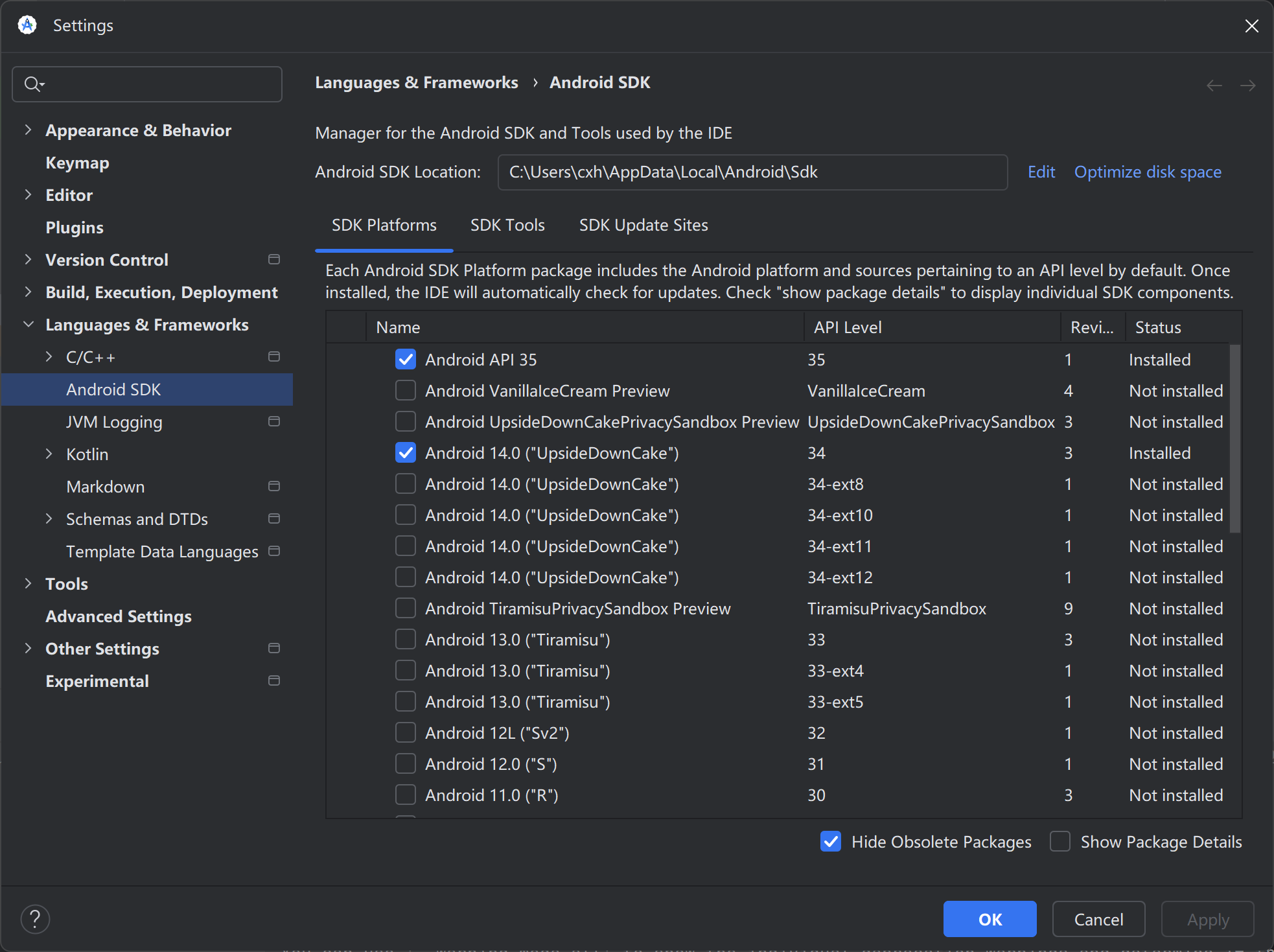
接受“android-sdk-license”和“intel-android-extra-license”

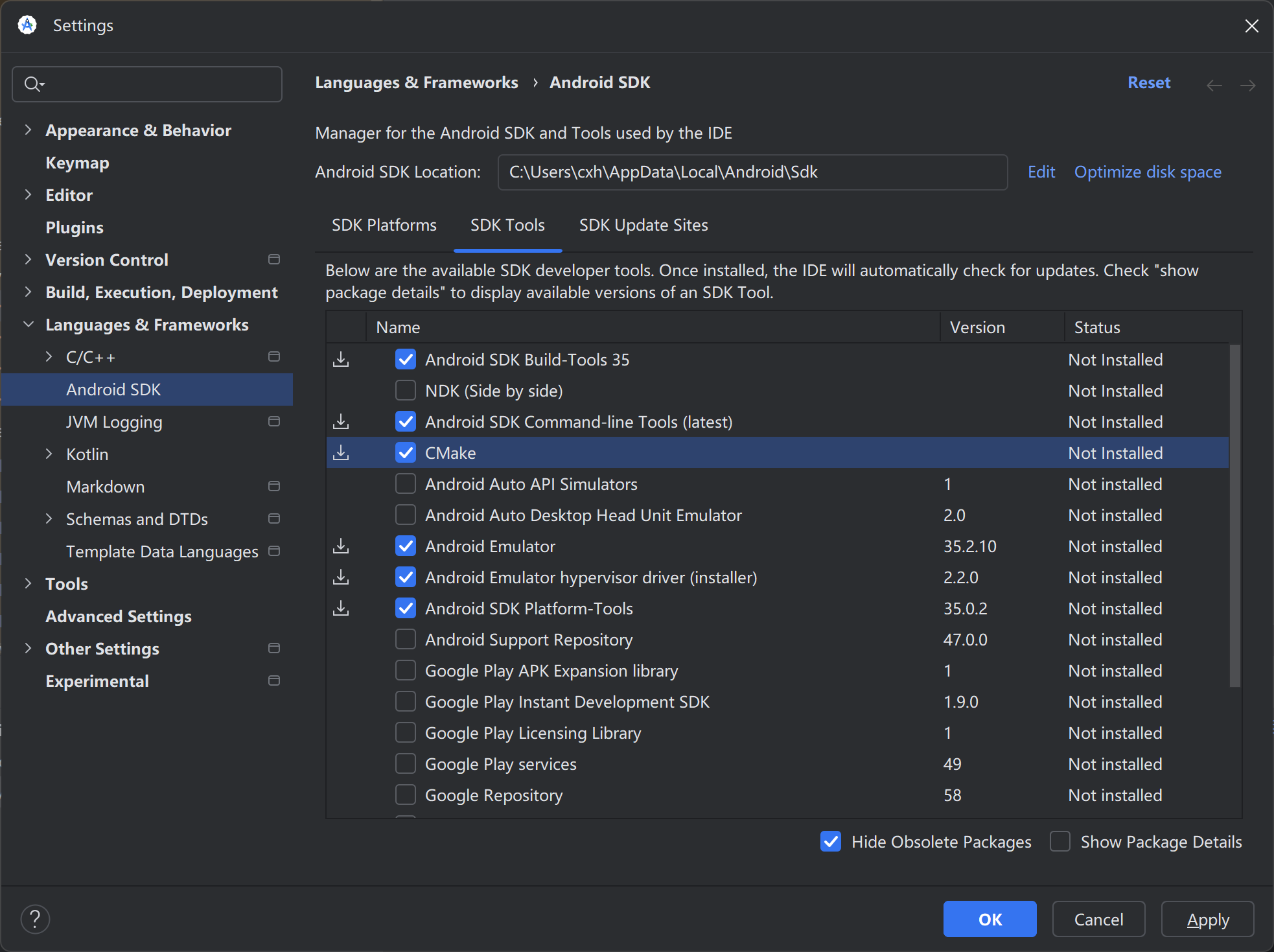


如果网络连接没有问题，则Android Studio安装完毕。

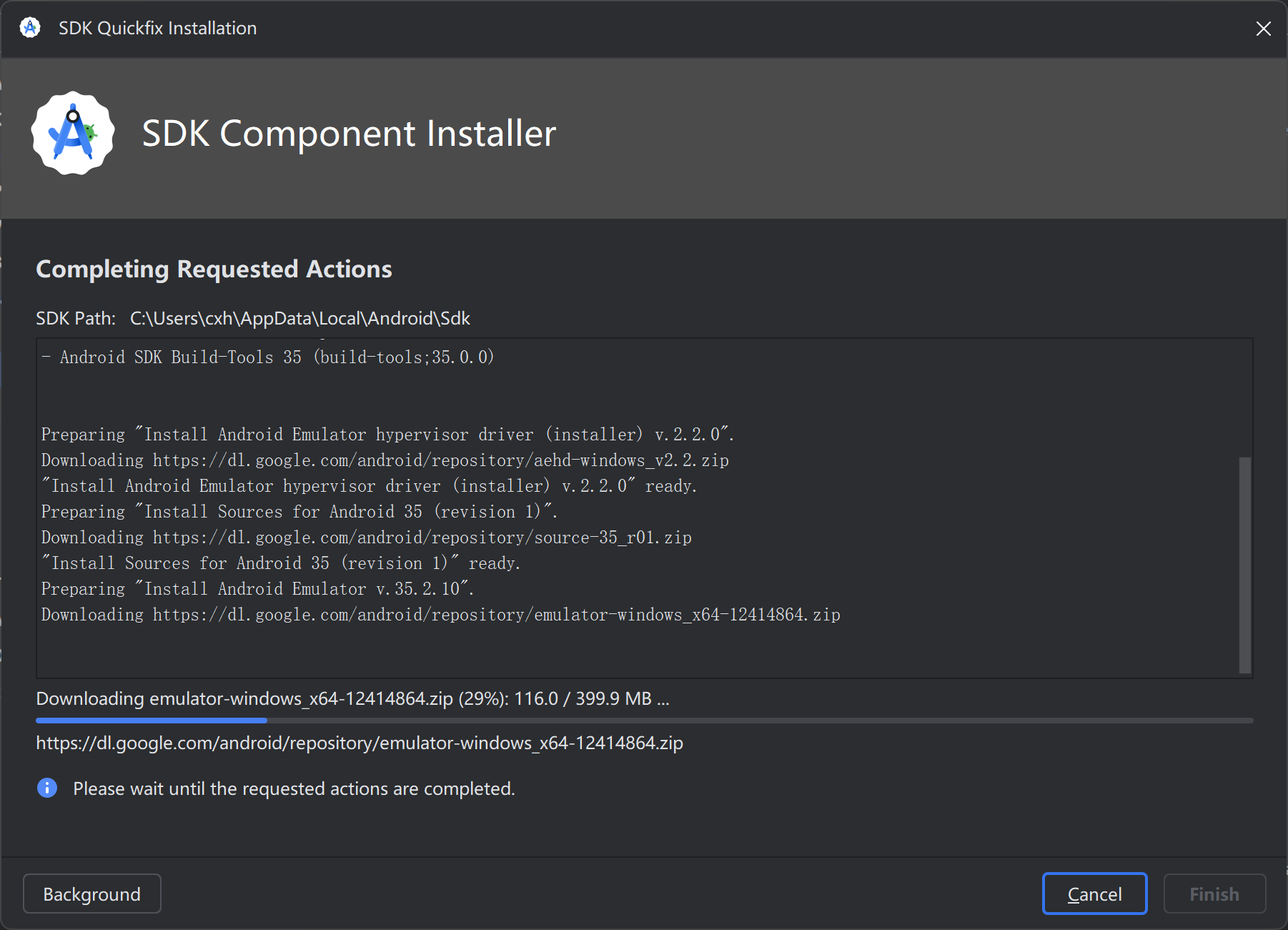
#### 手动安装Android SDK

如果网络连接有问题，Android SDK安装失败，则需要手动安装SDK。启动Android Studio后，新建一个Project，进入到主界面。主菜单“Tool”→“SDK Manager”打开Android SDK包管理器，勾选必要的包（SDK Platforms、SDK Tools），建议安装的包如下。





然后进入漫长的下载和安装等待过程（教育网内下载速度很慢）。



### 关于国内代理

如果所在的局域网直连<http://dl.google.com>有问题，需要设置国内代理。常用的国内代理：

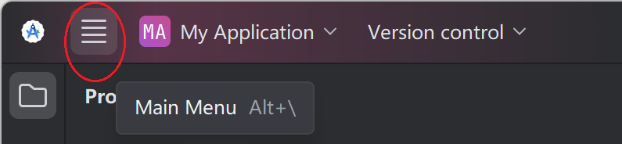
清华：mirrors.tuna.tsinghua.edu.cn 端口：80

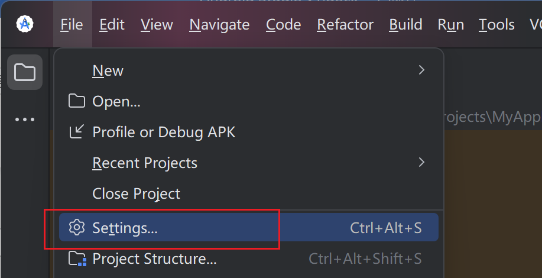
腾讯：mirrors.cloud.tencent.com 端口：80

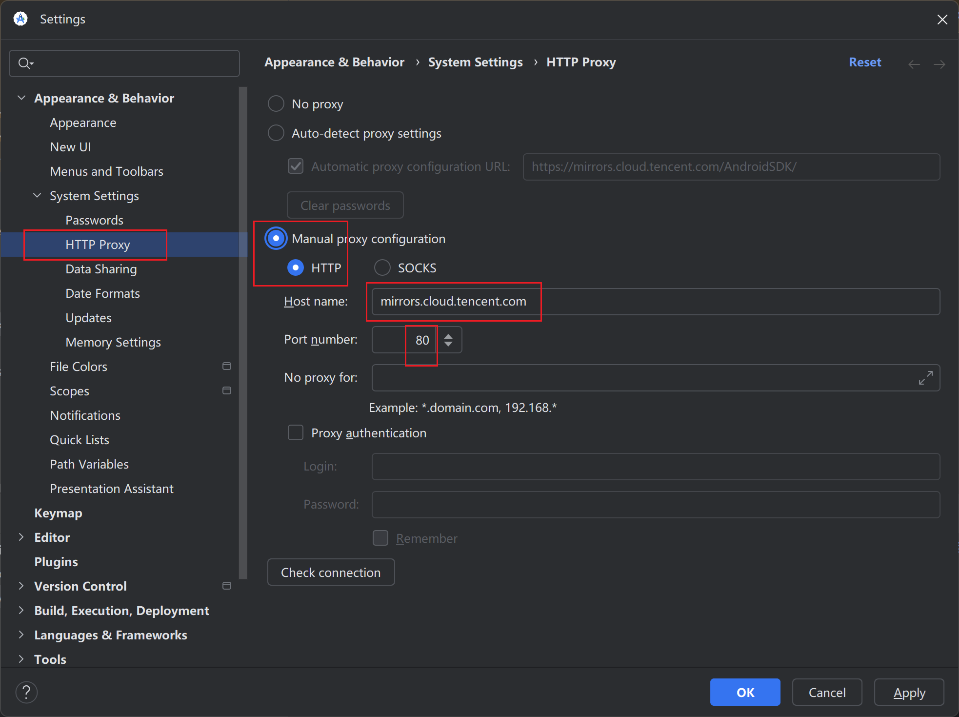
阿里：mirrors.aliyun.com 端口：80

在校园网内，可能不需要设置代理也可以正常连接<http://dl.google.com>，实测网络通出口选择移动网出口速度比较快。

主菜单“File”→“Setting”可以打开HTTP代理的配置界面。







笔者经验：Android Studio开发环境安装和配置过程中，网络连接一直是困扰程序员的一个问题。由于谷歌公司的一些服务并未在大陆地区开展，所以很多软件包的获取需要通过代理才可以。而校园网在教育网内，访问国内的一些代理可能网络也不稳定。实际配置环境的时候，可能有些软件包可以通过国内代理连接获取，而另外一些软件包需要直连谷歌网站才可以获取。掌握上述配置代理的方法后，不同时间段、不同软件包更新需求情形，可以在使用代理和不使用代理2种状态之间切换。

## 创建第一个Android Studio项目

### 基础知识：Project与Project的自动构建

#### 项目（Project）

Android Studio 中的项目包含为应用定义工作区的所有内容，包括源代码、资源、测试代码和 build 配置。使用Android Studio新建Project时，Android Studio 会为所有文件创建必要的结构。项目内的主要组件包括：模块、项目文件、项目结构设置。此处仅做简要介绍，

更多关于项目的描述请参阅 <https://developer.android.google.cn/studio/projects?hl=zh-cn>。

模块（Module）是源文件和构建设置的集合，可让您将项目划分为独立的功能单元。您的项目可以包含一个或多个模块，一个模块可以将另一个模块作为依赖项。可以独立构建、测试和调试每个模块。Android Studio 提供了几种不同类型的模块：（1）Android 应用模块。此类模块为应用的源代码、资源文件和应用级设置（例如模块级 build 文件和 Android 清单文件）提供容器。当您创建新项目时，默认应用模块会命名为“app”。（2）功能模块。此类模块表示应用中可利用 Play Feature Delivery 的模块化功能。例如，借助功能模块，您可以按需为用户提供应用的某些功能，或通过 Google Play 免安装体验技术以免安装形式为用户提供应用的某些功能。（3）库模块。此类模块为可重用代码提供容器，您可以将其作为依赖项用在其他应用模块中或将其导入其他项目中。

Android Studio可以用Project 视图来显示项目中的所有文件和目录，也可以用Android视图来显示项目文件。在Andoird视图下，所有文件被显示为不同的组：（1）manifests，包含 AndroidManifest.xml 文件。（2）java，包含 Kotlin[[4]](#footnote-4) 和 Java 源代码文件及JUnit 测试代码。（3）res，包含所有非代码资源（例如界面字符串和位图图像）。

#### 自动化构建（Build Automation）

在我们初学C语言时，C源程序经过预处理(Preprocessing)、编译(Compilation)、链接(Linking)等步骤。若使用gcc编译，可以使用一个或多个命令组合完成这个过程。例如，“gcc test1.c -o test1.exe”会根据C源程序“test1.c”生成名称为“test1.exe”的可执行文件。

当程序功能较为复杂（如包含多个C源文件、多个C源程序保存在不同的目录、调用了多个第三方库等情形）时，手工指定gcc参数列表较为繁琐且容易出错。此时，我们可以编写makefile来把多个操作写入单个makefile文件。makefile可以帮助我们按照顺序编译程序或者执行一些功能，它其实就是一种自动化构建工具（Build Automation）。

当然，我们也可以借助IDE[[5]](#footnote-5)来完成这个繁琐的过程。通常，IDE会通过项目（Project）来管理多个源文件、第三方库等系列生成可执行文件需要的资源。例如，Android Studio 中新建Project时，Android Studio 会为所有文件创建必要的结构，生成build有关的所有脚本，从而用户可以不用关注具体编译、连接的细节步骤。Android Studio中这些自动化构建依赖一种称之为Gradle的软件包来实现。

#### 关于自动化构建工具

通俗来说，自动化构建工具就是自动将源码生成为可执行文件的工具。常用的组建自动化工具就有Make、MS build、Ant、Gradle等。Android Studio借助Gradle来实现项目（Project）的自动化管理。

Gradle是一个基于Apache Ant和Apache Maven概念的项目自动化建构工具。Gradle脚本使用DSL[[6]](#footnote-6)来声明项目设置，早期版本的Android Studio使用Groove DSL，但高版本Android Studio中中谷歌推荐使用Kotlin DSL。谷歌将Gradle方式作为Android Studio项目管理的默认方式，使用android studio创建的项目会默认生成build.gradle文件作为默认构建。Gradle贯穿了项目的整个生命周期，包括编译、检查、测试、打包、部署。



#### Android Studio项目的Gradle配置文件

了解Gradle对于理解Android Studio创建项目、生成目标机上的可执行二进制文件等不同步骤非常重要。在Android Studio向导生成的项目中，会有多个Gradle配置文件，如图 2‑1所示。

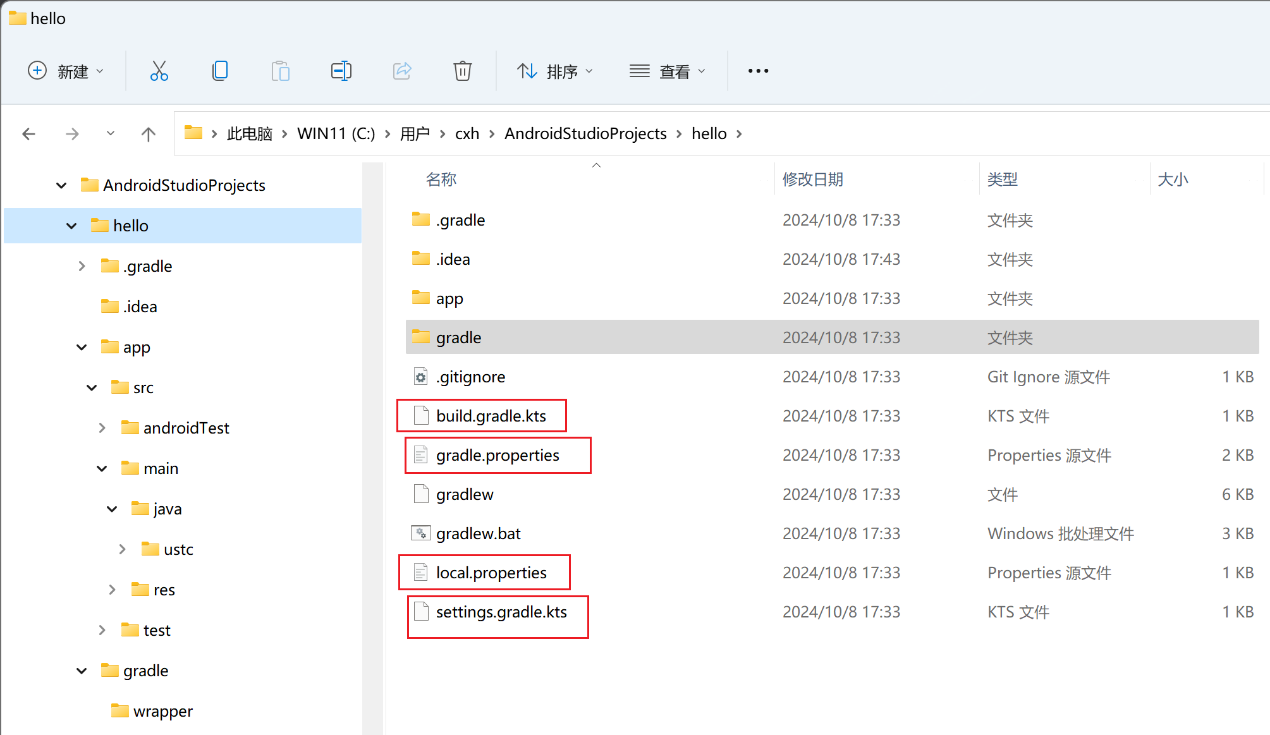


图 12‑1 向导生成Project目录中的Gradle配置文件

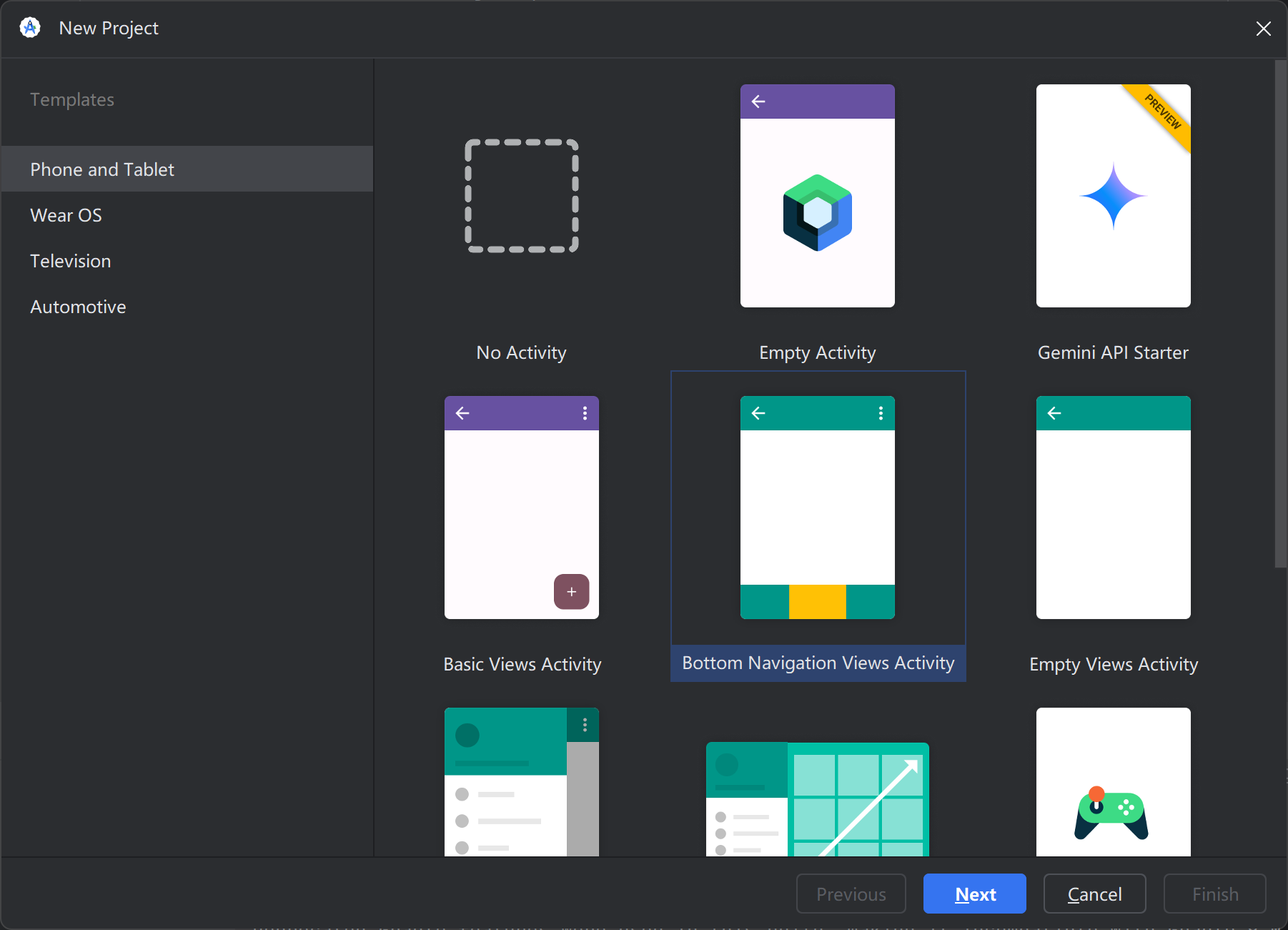
这些Gradle配置文件包括：

* build.gradle（若使用Kotlin DSL，则为build.gradle.kts）: 在app目录下，存在一个build.gradle文件，代表了app Module的构建脚本，它定义了应用于本模块的构建规则。项目根目录下也存在一个build.gradle文件，它代表整个项目的构建，定义适用于这个项目中所有模块的构建规则。
* gradle.properties: 从它的名字可以看出，这个文件中定义了一系列“属性”。实际上，这个文件中定义了一系列供build.gradle使用的常量，比如keystore的存储路径、keyalias等等。
* gradlew与gradlew.bat: gradlew为Linux下的shell脚本，gradlew.bat是Windows下的批处理文件。gradlew是gradle wrapper的缩写，也就是说它对gradle的命令进行了包装，比如我们进入到指定Module目录并执行“gradlew.bat assemble”即可完成对当前Module的构建（Windows系统下）。
* local.properties: 从名字就可以看出来，这个文件中定义了一些本地属性，比如SDK的路径。
* settings.gradle（若使用Kotlin DSL，则为settings.gradle.kts）: 假如我们的项目包含了不只一个Module时，我们想要一次性构建所有Module以完成整个项目的构建，这时我们需要用到这个文件。比如我们的项目包含了ModuleA和ModuleB这两个模块，则这个文件中会包含这样的语句：include ':ModuleA', ':ModuleB'。

### 构建一个“Bottom Navigation Views Activity”类型的Project

#### 使用图形化向导生成项目文件

主菜单“File”→“New”→“New Project”可以打开图形化的向导来辅助项目（Project）的建立。建议第一个项目选择如下所示的“Bottom Navigation Views Activity”，所创建的项目生成的APP程序（\*.apk安装后在手机上生成的APP）的界面底部会有3个按钮，点击按钮会显示一些提示文字。



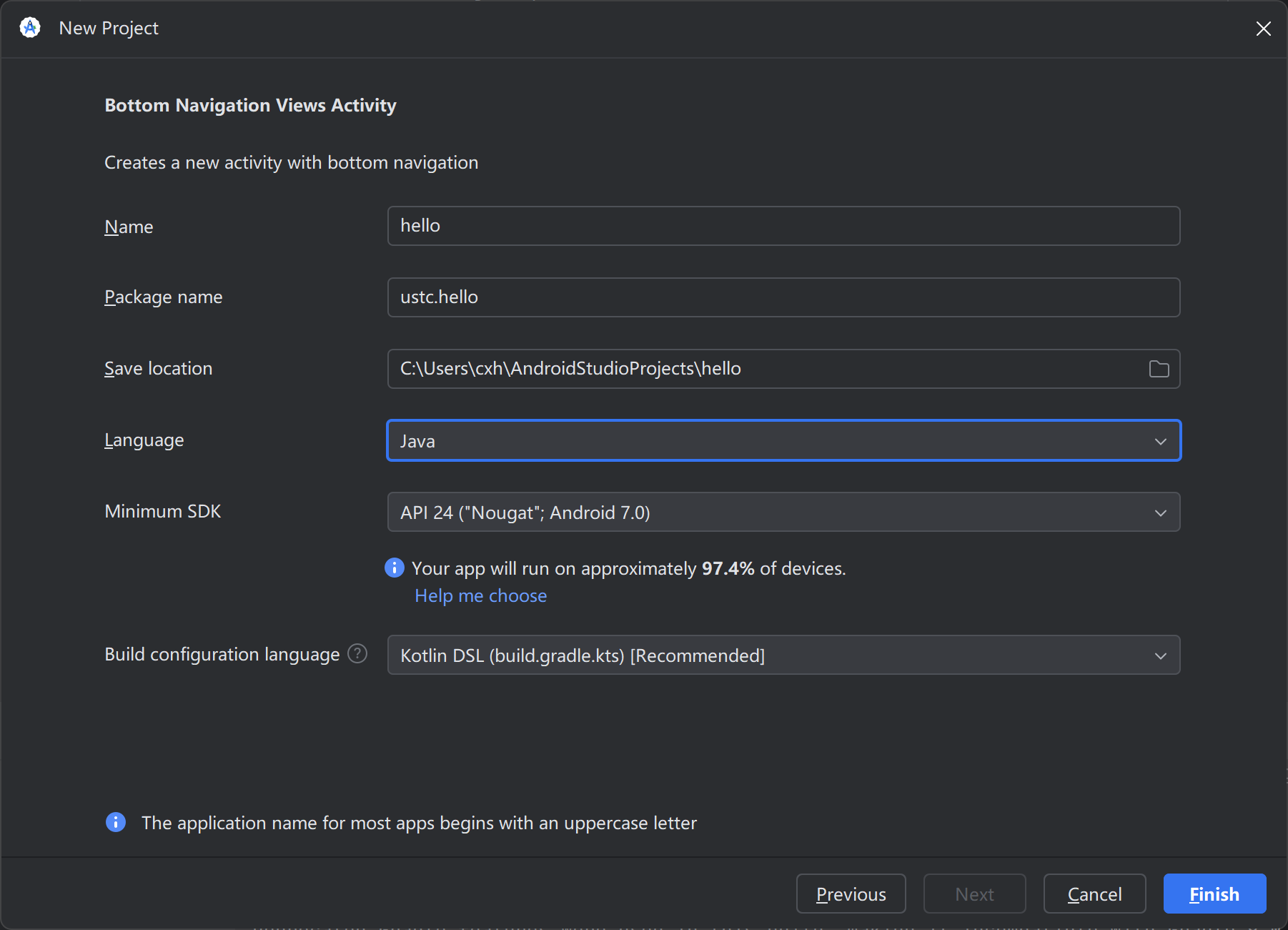
选定界面风格后，建议选择/设置如下。

Name，Project的名称

Package name，所生成软件包的名称

Language，建议选择java，另一个选项是Kotlin[[7]](#footnote-7)

Build configuration language，建议选用默认的Kotlin DSL，另一个选项是Groove DSL[[8]](#footnote-8)



Android Studio的图形化向导会生成项目所需要的一系列文件（java源文件、UI界面资源相关文件、Gradle配置文件等），如图 12‑2所示。其中：

* java源文件所在目录为“\AndroidStudioProjects\hello\app\src\main\java”。
* UI资源文件所在目录为“\AndroidStudioProjects\hello\app\src\main\res\layout”。
* 几个Gradle配置文件分布在目录“\AndroidStudioProjects\hello”和“\AndroidStudioProjects\hello\gradle\wrapper”。

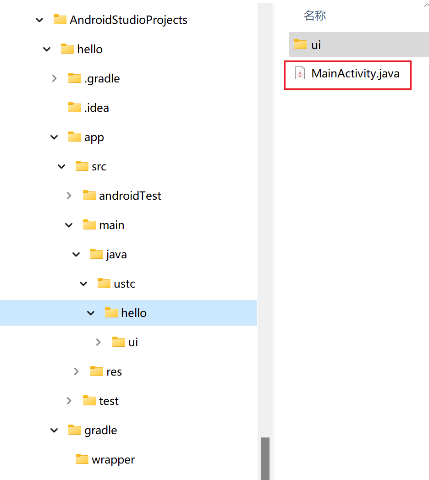
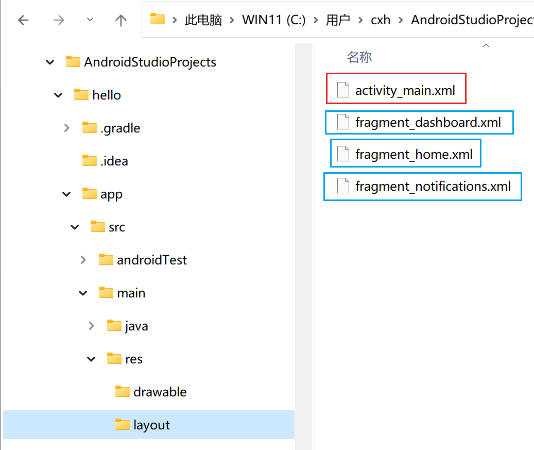
 

图 12‑2 向导生成的java源文件（左，MainActivity.java）和UI界面描述文件（右，\*.xml）

上述生成的文件可以在Android Studio的Android视图下查看，例如图 12‑3所示的树形结构把各个不同用途的文件按照分组予以显示。

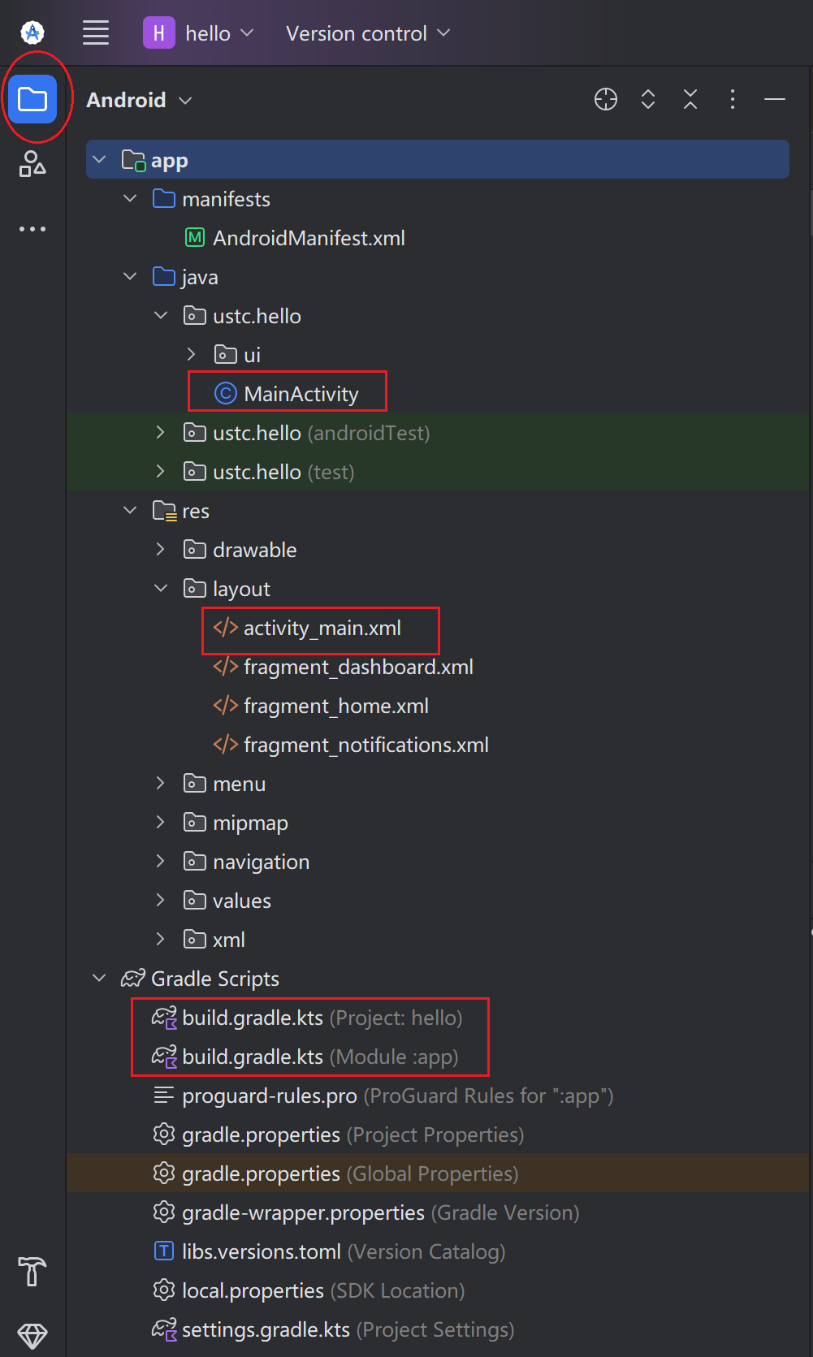


图 12‑3 Android Studio中查看项目的文件

#### 简单了解Android Studio中的UI设计

Android Studio的图形化向导完成所有文件的生成后，会打开UI界面描述文件（activity\_main.xml），熟悉XML[[9]](#footnote-9)句法规则（syntax rule）的同学可以切换到纯文本状态直接修改xml文件内容来实现界面的调整。在UI界面较为简单的情形，一个activity\_main.xml就可以描述所有界面相关信息，如本课程实验中的示例都仅使用了一个XML文件。而图 12‑4所示UI，共有4个XML文件，主界面对应activity\_main.xml，而3个按钮分别对应了3个XML文件。

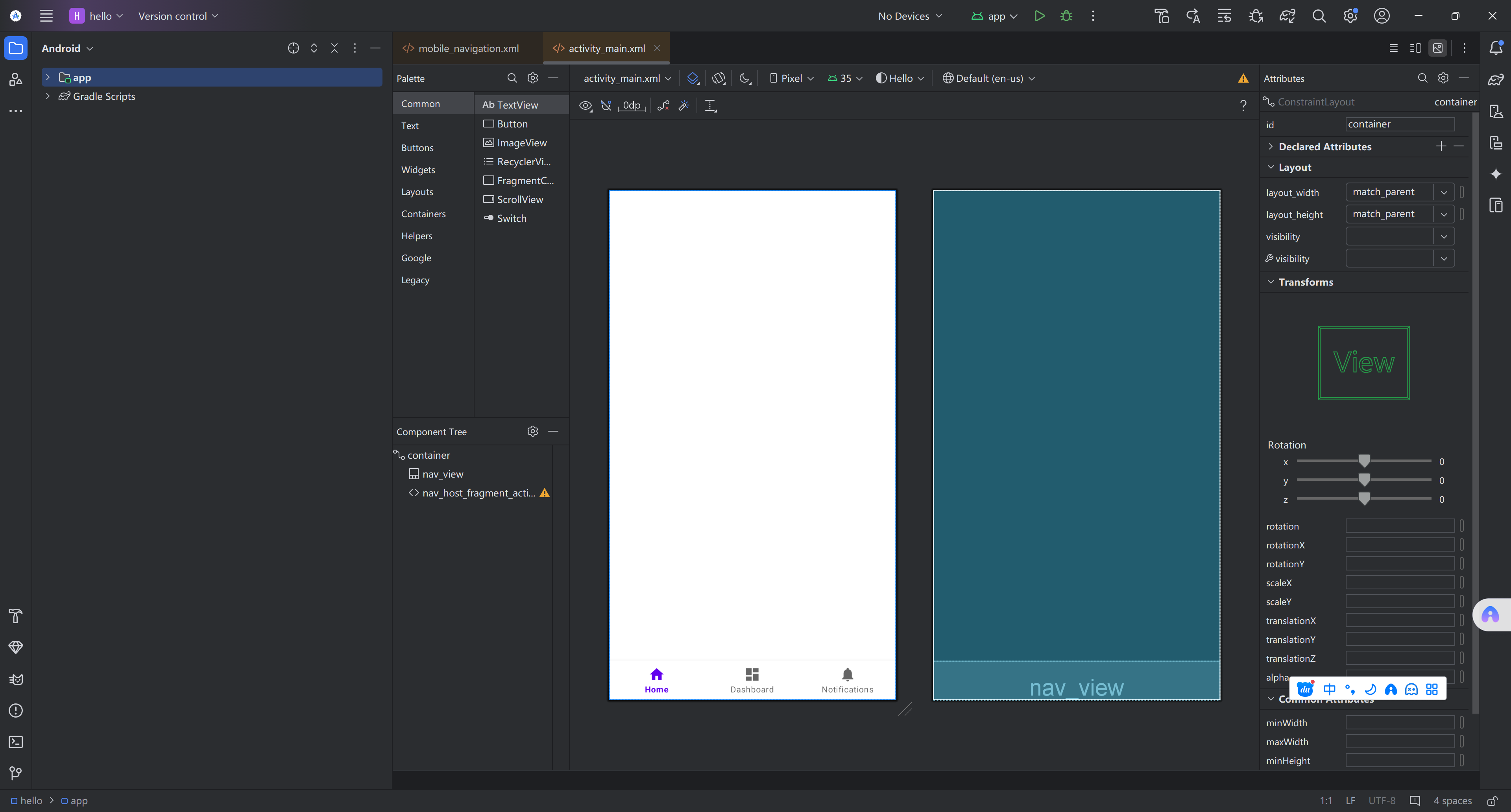


图 12‑4 Android Studio中UI编辑界面

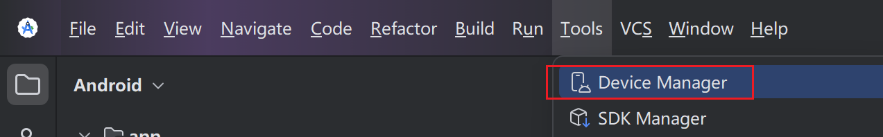
Android Studio的图形化界面及菜单功能的详细介绍，可参阅：

Get to know the Android Studio UI  |  Android Developers (google.cn)

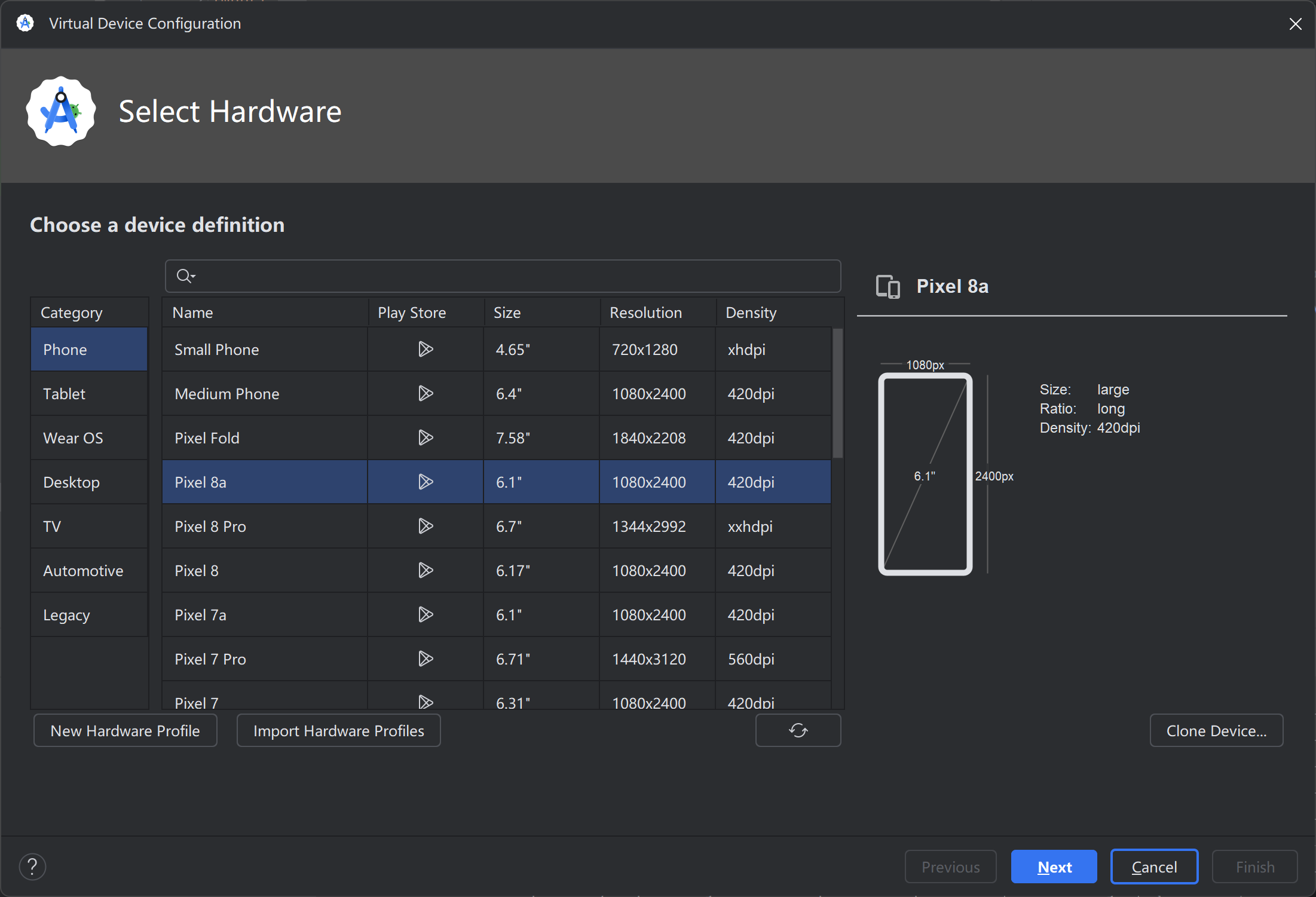
<https://developer.android.google.cn/studio/intro/user-interface>

#### 在虚拟设备上运行程序

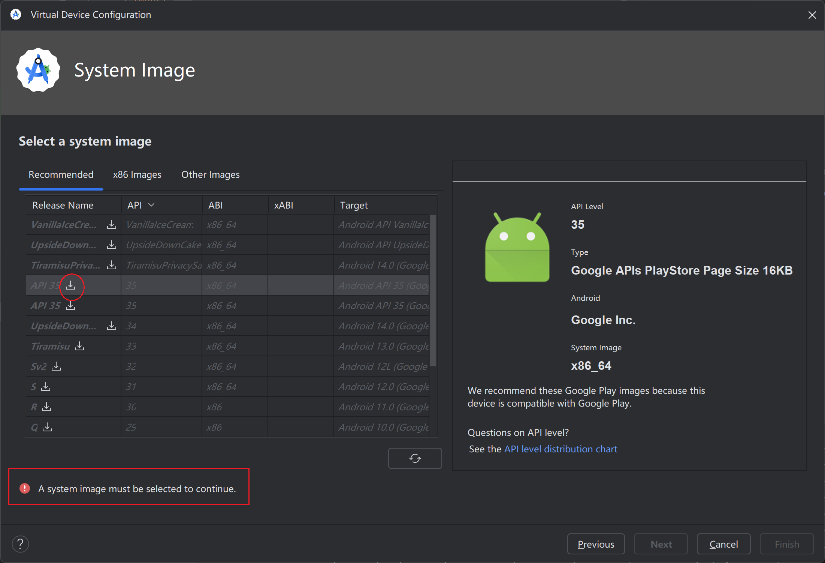
由于Android Studio开发的程序需要在手机、平板等终端设备上使用，往往先借助虚拟设备对程序运行的效果和性能进行调试。故而我们在进一步build project之前，先创建虚拟设备。打开“Device Manager”，根据“Device Manager”向导增加设备。



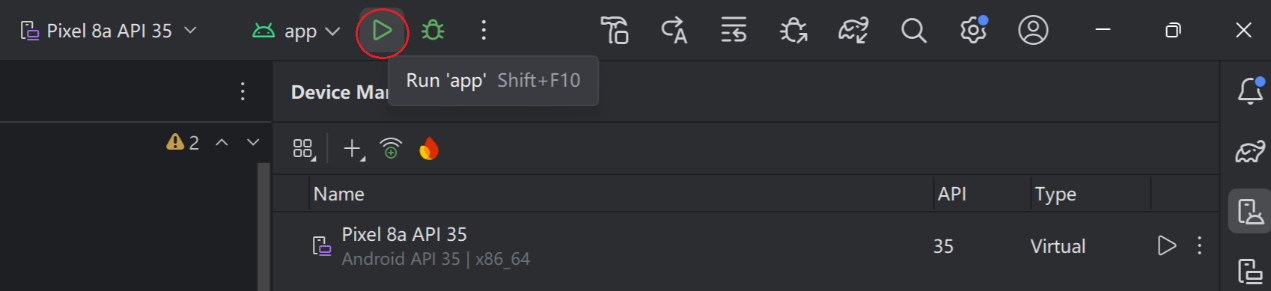
例如，创建“Pixel 8a”型号的虚拟手机。



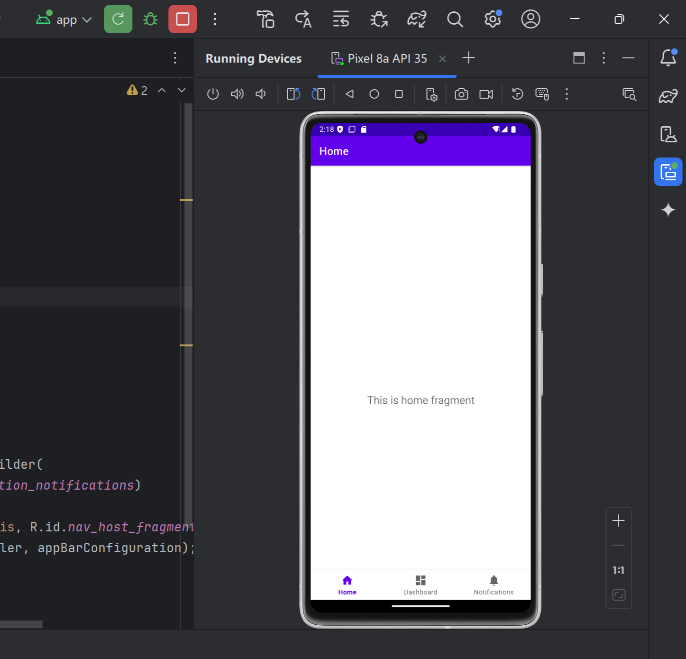
如果提示缺少system image，需要下载所选择虚拟机需要的软件包。



所需要的软件包下载完成后，按照向导提示就可以创建“Pixel 8a”型号的虚拟手机。这时候可以直接点击如下图所示 Run 'app' 按钮开始项目的build，Gradle工具会自动下载build过程所需要的软件工具，这个过程中如果缺少某些软件包，根据提示安装即可。



Build结束后，会启动虚拟设备并将程序在虚拟设备上运行。如下图所示，我们所创建“Bottom Navigation Views Activity”类型的Project生成的APP程序在虚拟手机上显示了屏幕底部有3个按钮的界面，点击按钮，会显示一些提示文字。至此，我们已经成功地创建并运行了一个Android Studio项目。



Android官网也提供了Android Studio的入门教程，可参阅：

构建首个应用  |  Get started  |  Android Developers (google.cn)

<https://developer.android.google.cn/get-started/overview?hl=zh-cn>

### 进阶：UI界面图元和代码的关联是如何实现的

笔者注：在我校的程序设计课程学习中，对C语言的语法、用C语言实现算法思想会较为侧重，但涉及图形化应用程序的内容罕见。且因为程序设计课程开设年级低，缺少了程序和计算机硬件的关联、程序的UI设计、专业领域程序与专业课关联等环节。此外，程序设计仅涉及编程的基础知识，难以全面体现实际应用软件开发工具的变化趋势。本小节针对图形化用户界面中，界面上图元（按钮、输入框等）与代码的关联稍作说明。

在进一步build project之前，先来了解一下界面上各个图形化控件（按钮、文本框等）的操作是如何与java代码关联的。建议同学稍微了解一些XML的句法规则后再阅读本小节的内容。

描述UI的XML文件中，每个图形化的控件都有id。例如，界面底部放置3个按钮的矩形区域图元对应的id是nav\_view（android:id="@+id/nav\_view"）。如果在java源代码中要获取/设置这个图元的状态，可以通过对这个id的引用来实现（R.id.*nav\_view*）。

#### 主界面描述文件activity\_main.xml

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 android:id="@+id/container"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:paddingTop="?attr/actionBarSize">  
  
 <com.google.android.material.bottomnavigation.BottomNavigationView  
 android:id="@+id/nav\_view"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="0dp"  
 android:layout\_marginEnd="0dp"  
 android:background="?android:attr/windowBackground"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:menu="@menu/bottom\_nav\_menu" />  
  
 <fragment  
 android:id="@+id/nav\_host\_fragment\_activity\_main"  
 android:name="androidx.navigation.fragment.NavHostFragment"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 app:defaultNavHost="true"  
 app:layout\_constraintBottom\_toTopOf="@id/nav\_view"  
 app:layout\_constraintLeft\_toLeftOf="parent"  
 app:layout\_constraintRight\_toRightOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:navGraph="@navigation/mobile\_navigation" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

可以在一个XML中引用其他的XML文件。例如，主界面底部放置了id为nav\_host\_fragment\_activity\_main的图元包含了mobile\_navigation.xml文件的内容。而mobile\_navigation.xml文件则描述了3个按钮的图元信息。

#### 图元nav\_host\_fragment\_activity\_main的描述文件mobile\_navigation.xml

<?xml version="1.0" encoding="utf-8"?>  
<navigation xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/mobile\_navigation"  
 app:startDestination="@+id/navigation\_home">  
  
 <fragment  
 android:id="@+id/navigation\_home"  
 android:name="ustc.hello.ui.home.HomeFragment"  
 android:label="@string/title\_home"  
 tools:layout="@layout/fragment\_home" />  
  
 <fragment  
 android:id="@+id/navigation\_dashboard"  
 android:name="ustc.hello.ui.dashboard.DashboardFragment"  
 android:label="@string/title\_dashboard"  
 tools:layout="@layout/fragment\_dashboard" />  
  
 <fragment  
 android:id="@+id/navigation\_notifications"  
 android:name="ustc.hello.ui.notifications.NotificationsFragment"  
 android:label="@string/title\_notifications"  
 tools:layout="@layout/fragment\_notifications" />  
</navigation>

#### 与界面对应的处理代码MainActivity.java

public class MainActivity extends AppCompatActivity {  
  
 private ActivityMainBinding binding;  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
  
 binding = ActivityMainBinding.*inflate*(getLayoutInflater());  
 setContentView(binding.getRoot());  
  
 BottomNavigationView navView = findViewById(R.id.*nav\_view*);  
 // Passing each menu ID as a set of Ids because each  
 // menu should be considered as top level destinations.  
 AppBarConfiguration appBarConfiguration = new AppBarConfiguration.Builder(  
 R.id.*navigation\_home*, R.id.*navigation\_dashboard*, R.id.*navigation\_notifications*)  
 .build();  
 NavController navController = Navigation.*findNavController*(this, R.id.*nav\_host\_fragment\_activity\_main*);  
 NavigationUI.*setupActionBarWithNavController*(this, navController, appBarConfiguration);  
 NavigationUI.*setupWithNavController*(binding.navView, navController);  
 }  
  
}

### 进阶：Gradle及build 配置

笔者注：在Android Studio的项目（Project）的构建过程中，通过构建工具软件Gradle来简化软件设计过程，而Gradle本身作为一个独立的软件工具是需要进行配置的。对于初学者而言，Gradle的配置过程并不容易理解。本小节尝试简要介绍一下Gradle在build过程中发挥作用的过程。

Android Studio 使用高级构建工具包 Gradle 来自动执行和管理构建流程，同时也允许自行指定灵活的 build 配置。Android Gradle 插件（Plugin）与该构建工具包搭配使用，提供专用于构建和测试 Android 应用的流程和可配置设置。

Gradle 和 Android Gradle 插件独立于 Android Studio 运行。这意味着，您可以在 Android Studio 内、计算机上的命令行或未安装 Android Studio 的计算机（如持续集成服务器）上构建 Android 应用。关于配置 build的详细介绍可参阅：

<https://developer.android.google.cn/build?hl=bg>

#### build 配置文件

创建自定义 build 配置需要对一个或多个 build 配置进行更改，可以使用Groovy DSL或Kotlin DSL来构建这些配置文件。使用向导创建一个新的Project时，Android Studio 会自动创建这些配置文件。项目文件结构的布局如下：

└── MyApp/ # Project

├── gradle/

│ └── wrapper/

│ └── gradle-wrapper.properties

├── build.gradle(.kts)

├── settings.gradle(.kts)

└── app/ # Module

│ ├── build.gradle(.kts)

│ ├── build/

│ ├── libs/

│ └── src/

│ └── main/ # Source set

│ ├── java/

│ │ └── com.example.myapp

│ ├── res/

│ │ ├── drawable/

│ │ ├── values/

│ │ └── ...

│ └── AndroidManifest.xml

#### Gradle 封装容器（Gradle Wrapper）和gradle-wrapper.properties文件

Gradle 封装容器 (gradlew) 是一个用于下载和启动 Gradle 本身的源代码。Gradle Wrapper的作用是简化Gradle本身的安装、部署。不同版本的项目可能需要不同版本的Gradle，手工部署的话比较麻烦，而且可能产生冲突，所以需要Gradle Wrapper帮你搞定这些事情。

Gradlew会根据配置文件“gradle/wrapper/gradle-wrapper.properties”的内容来下载指定版本的Gradle并运行。“gradle/wrapper/gradle-wrapper.properties”文件格式如下。

distributionBase=GRADLE\_USER\_HOME

distributionPath=wrapper/dists

distributionUrl=https\://services.gradle.org/distributions/gradle-8.0-bin.zip

zipStoreBase=GRADLE\_USER\_HOME

zipStorePath=wrapper/dists

#### Gradle 设置文件

settings.gradle.kts 文件（适用于 Kotlin DSL）或 settings.gradle 文件（对于 Groovy DSL）位于根目录下项目目录中。此设置文件定义项目所使用的软件仓库（或称代码库）。如下为一个settings.gradle.kts文件的示例。

pluginManagement {

repositories {

google {

content {

includeGroupByRegex("com\\.android.\*")

includeGroupByRegex("com\\.google.\*")

includeGroupByRegex("androidx.\*")

}

}

mavenCentral()

gradlePluginPortal()

}

}

dependencyResolutionManagement {

repositoriesMode.set(RepositoriesMode.FAIL\_ON\_PROJECT\_REPOS)

repositories {

google()

mavenCentral()

}

}

上述文件中pluginManagement部分定义了插件（Plugins，全称为Android **Plugin** for Gradle）。插件是扩展Gradle功能的组件，用于自动化构建过程中的各种任务，如编译、打包、测试等。这些插件存放在网络仓库（remote repositories）中。

* 其中repositories {}部分定义了3个远程仓库：Gradle Plugin Portal、 Google's Maven repository、Maven Central Repository，即，Gradle会从这3个仓库获取所需要的插件模块，仓库的列出顺序决定了 Gradle 在代码库中搜索每个项目依赖项的顺序。
* includeGroupByRegex 函数指示按照正则表达式包含特定的模块或组。文件中 includeGroupByRegex("com\\.android.\*") 指示以“com.android”开头的模块。

上述文件中dependencyResolutionManagement部分定义了依赖（Dependencies）。依赖是指项目中使用的外部库或模块，通常是从远程仓库下载的JAR文件或其他类型的包。

* 其中repositories {}部分定义了2个远程仓库： Google's Maven repository、Maven Central Repository，即，Gradle会从这2个仓库获取所需要的外部软件模块。
* repositoriesMode.set(RepositoriesMode.FAIL\_ON\_PROJECT\_REPOS)的含义是解析依赖时，只能使用本脚本块中的仓库，不能使用 Module 中的依赖。

#### 顶层 build 文件

顶级 build.gradle.kts 文件（适用于 Kotlin DSL）或 build.gradle 文件（对于 Groovy DSL）位于根目录下 项目目录中。它通常定义了常用版本的插件， 不同模块。如下为一个示例。

plugins {

alias(libs.plugins.android.application) apply false

}

在这个build.gradle.kts示例文件中，alias(libs.plugins.android.application) 引入了一个名为 androidApplication 的插件。apply false：这个部分表示插件在引入时不会立即应用到项目中。通常，插件在被引入时会自动应用到项目中，但是通过将 apply 参数设置为 false，可以让插件在被引入后需要手动调用 apply 方法才能生效。

#### 模块级 build 文件

模块级 build.gradle.kts（适用于 Kotlin DSL）或 build.gradle 文件（适用于 Groovy DSL）位于 project/module/ 目录中。用于为其所在的特定模块配置 build 设置。正在配置 这些构建设置可让您提供自定义打包选项，例如 其他构建类型和产品变种，并替换 main/ 应用清单或顶级构建脚本。如下是一个示例。

plugins **{** alias(*libs*.*plugins*.*android*.*application*)  
**}***android* **{** namespace = "ustc.hello"  
 compileSdk = 34  
  
 defaultConfig **{** applicationId = "ustc.hello"  
 minSdk = 24  
 targetSdk = 34  
 versionCode = 1  
 versionName = "1.0"  
  
 testInstrumentationRunner = "androidx.test.runner.AndroidJUnitRunner"  
 **}** buildTypes **{** *release* **{** isMinifyEnabled = false  
 proguardFiles(  
 getDefaultProguardFile("proguard-android-optimize.txt"),  
 "proguard-rules.pro"  
 )  
 **}  
 }** compileOptions **{** sourceCompatibility = JavaVersion.*VERSION\_1\_8* targetCompatibility = JavaVersion.*VERSION\_1\_8* **}** buildFeatures **{** viewBinding = true  
 **}  
}***dependencies* **{** *implementation*(*libs*.*appcompat*)  
 *implementation*(*libs*.*material*)  
 *testImplementation*(*libs*.*junit*)  
 *androidTestImplementation*(*libs*.*ext*.*junit*)  
 *androidTestImplementation*(*libs*.*espresso*.*core*)  
**}**

上述示例文件中，plugins{}定义模块所需要的插件。

上述示例文件中，android{}定义了本模块build的参数。

* 其中defaultConfig {}定义了模块build时的默认参数。
* 其中buildTypes {}定义了build的类型，release还是debug，是否有签名。
* 其中compileOptions {}指定生成的java字节码版本。
* 其中buildFeatures {}指定一些Android Project的特性是否使能。如，使能View Binding，可以更轻松地编写与视图交互的代码。在模块中启用View Binding后，它将为该模块中存在的每个XML布局文件生成一个绑定类。绑定类的实例包含对在相应布局中具有ID的所有视图的直接引用。

上述示例文件中，dependencies {}定义了本模块中build时需要的依赖。

### 进阶：Android Studio各相关软件包的兼容性

笔者注：初学者往往是从运行各种示例开始学习的。然而，经常遇到示例无法正常build的情况，Android Studio给出的错误提示也难以理解。根据笔者经验，很多示例都是在特定Android SDK版本、JDK版本、Gradle版本下设计并build通过的，而Android Studio的版本更新很快，往往学习者所用的软件环境和示例撰写者的软件环境有差异，进而导致示例无法正常build。故而专门整理了这个小节的内容，简略介绍一下初学者在运行示例的过程中可能遇到的软件包版本兼容性。

Android build 中的 Java 版本  |  Android Studio  |  Android Developers (google.cn)

<https://developer.android.google.cn/build/jdks?hl=bg#jdk-config-in-studio>

如图 12‑5所示，在Android Studio开发过程中，从Java源代码生成JVM的bytecode、Gradle运行、测试（Test）等各个环节都会使用到JDK。在同一台计算机上使用Android Studio，可能会遇到不同示例对JDK版本有不同的要求，甚至同一个示例中由于Gradle版本升级导致java源代码解析所需要的的JDK版本和Gradle解析所需要的JDK不同。这些不同软件包之间会产生系列兼容性的问题，往往给初学者带来困惑。

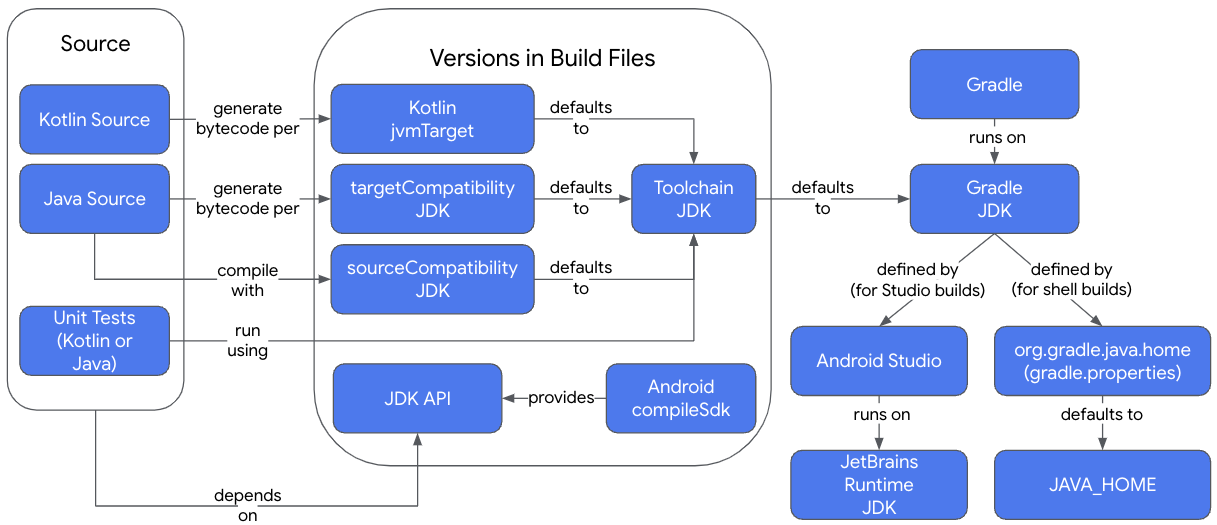


图 12‑5 Android Studio开发过程中多处使用JDK

#### Java源代码、Java字节码、JDK版本

Android build 中的 Java 版本  |  Android Studio  |  Android Developers (google.cn)

<https://developer.android.google.cn/build/jdks?hl=bg#toolchain>

可以在build.gradle配置文件中指定java工具链版本（JDK的版本），如

java {

toolchain {

languageVersion = JavaLanguageVersion.of(17)

}

}

可以在build.gradle配置文件中指定编译Java源文件时的JDK版本（即哪些 Java 语言功能 在编译 Java 源代码期间可用），如

android {

compileOptions {

sourceCompatibility JavaVersion.VERSION\_17

}

}

可以在build.gradle配置文件中指定为Java或Kotlin 生成字节码时使用的格式版本，如

android {

compileOptions {

targetCompatibility JavaVersion.VERSION\_17

}

kotlinOptions {

jvmTarget '17'

}

}

#### Gradle版本兼容性

##### Gradle版本与JDK版本的兼容性

Compatibility Matrix (gradle.org)

<https://docs.gradle.org/current/userguide/compatibility.html#:~:text=Gradle%20plugins%20written%20in%20Groovy%20must%20use%20Groovy%203.x%20for>

表 12‑1 JDK与Gradle版本的兼容性

|  |  |  |
| --- | --- | --- |
| Java version | Support for toolchains | Support for running Gradle |
| 8 | N/A | 2.0 |
| 9 | N/A | 4.3 |
| 10 | N/A | 4.7 |
| 11 | N/A | 5.0 |
| 12 | N/A | 5.4 |
| 13 | N/A | 6.0 |
| 14 | N/A | 6.3 |
| 15 | 6.7 | 6.7 |
| 16 | 7.0 | 7.0 |
| 17 | 7.3 | 7.3 |
| 18 | 7.5 | 7.5 |
| 19 | 7.6 | 7.6 |
| 20 | 8.1 | 8.3 |
| 21 | 8.4 | 8.5 |
| 22 | 8.7 | 8.8 |
| 23 | 8.10 | 8.10 |
| 24 | N/A | N/A |

##### Gradle版本与Kotlin版本的兼容性

Compatibility Matrix (gradle.org)

<https://docs.gradle.org/current/userguide/compatibility.html#:~:text=Gradle%20plugins%20written%20in%20Groovy%20must%20use%20Groovy%203.x%20for>

表 12‑2 Kotlin与Gradle版本的兼容性

|  |  |  |
| --- | --- | --- |
| Embedded Kotlin version | Minimum Gradle version | Kotlin Language version |
| 1.3.10 | 5.0 | 1.3 |
| 1.3.11 | 5.1 | 1.3 |
| 1.3.20 | 5.2 | 1.3 |
| 1.3.21 | 5.3 | 1.3 |
| 1.3.31 | 5.5 | 1.3 |
| 1.3.41 | 5.6 | 1.3 |
| 1.3.50 | 6.0 | 1.3 |
| 1.3.61 | 6.1 | 1.3 |
| 1.3.70 | 6.3 | 1.3 |
| 1.3.71 | 6.4 | 1.3 |
| 1.3.72 | 6.5 | 1.3 |
| 1.4.20 | 6.8 | 1.3 |
| 1.4.31 | 7.0 | 1.4 |
| 1.5.21 | 7.2 | 1.4 |
| 1.5.31 | 7.3 | 1.4 |
| 1.6.21 | 7.5 | 1.4 |
| 1.7.10 | 7.6 | 1.4 |
| 1.8.10 | 8.0 | 1.8 |
| 1.8.20 | 8.2 | 1.8 |
| 1.9.0 | 8.3 | 1.8 |
| 1.9.10 | 8.4 | 1.8 |
| 1.9.20 | 8.5 | 1.8 |
| 1.9.22 | 8.7 | 1.8 |
| 1.9.23 | 8.9 | 1.8 |
| 1.9.24 | 8.10 | 1.8 |

##### Gradle版本与AGP(Android Gradle plugin)版本的兼容性

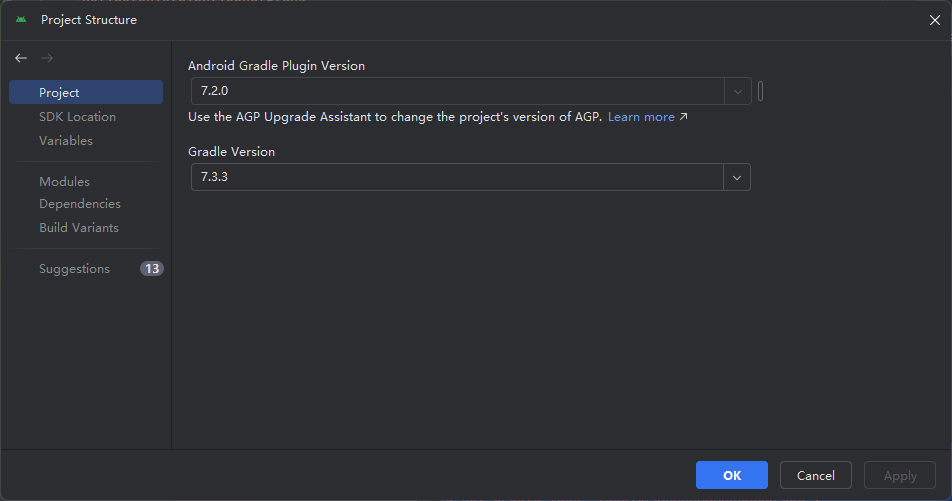
Gradle、Gradle plugin 基础认知\_android gradle plugin-CSDN博客

<https://blog.csdn.net/qq_20451879/article/details/85138675>

表 12‑3 AGP(Android Gradle plugin)与Gradle版本的兼容性

|  |  |
| --- | --- |
| 插件版本 | 所需的最低 Gradle 版本 |
| 8.0 | 8.0 |
| 7.4 | 7.5 |
| 7.3 | 7.4 |
| 7.2 | 7.3.3 |
| 7.1 | 7.2 |
| 7.0 | 7.0 |
| 4.2.0+ | 6.7.1 |

Android Studio中可以通过“Project Structure”查看Plugin版本和Gradle版本



# 附录：历史上开设过的实验项目

<http://staff.ustc.edu.cn/~network/mmt/>

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **实验名称** | **范例** | **内容分类** | **实验类别** |
| 实验一 | [MCAI1.0演示](http://staff.ustc.edu.cn/~network/mmt/mm_mcai.pdf) |  | 多媒体素材操作 | 演示型 |
| 实验二 | [动画制作](http://staff.ustc.edu.cn/~network/mmt/mm_material.pdf) |  | 多媒体素材制作 | 设计型 |
| 实验三 | [视频编辑](http://staff.ustc.edu.cn/~network/mmt/mm_premier.pdf) | [下载](http://staff.ustc.edu.cn/~network/mmt/mm_premier_eg.zip) | 多媒体素材制作 | 设计型 |
| 实验四 | [MCI接口编程](http://staff.ustc.edu.cn/~network/mmt/mm_mci.pdf) | [下载](http://staff.ustc.edu.cn/~network/mmt/test.rar) | 多媒体编程基础 | 设计型 |
| 实验五 | [WAVE文件格式分析](http://staff.ustc.edu.cn/~network/mmt/mm_wave.pdf) | [下载](http://staff.ustc.edu.cn/~network/mmt/sb2.zip) | 多媒体编程基础 | 设计型 |
| 实验六 | [基于API的视频卡编程](http://staff.ustc.edu.cn/~network/mmt/mm_api_video.pdf) | [下载](http://staff.ustc.edu.cn/~network/mmt/VideoSample.zip) | 多媒体编程基础 | 设计型 |
| 实验七 | [基于SDK的视频卡编程](http://staff.ustc.edu.cn/~network/mmt/mm_sdk_video.pdf) | [下载](http://staff.ustc.edu.cn/~network/mmt/mm_sdk_video_eg.zip) | 多媒体编程基础 | 设计型 |
| 实验八 | [基于DirectX的编程](http://staff.ustc.edu.cn/~network/mmt/mm_directx.pdf) | [下载](http://staff.ustc.edu.cn/~network/mmt/mm_directx_eg.zip) | 多媒体编程基础 | 设计型 |
| 实验九 | [JPEG静图像压缩](http://staff.ustc.edu.cn/~network/mmt/mm_jpeg_20210116.pdf) | [下载](http://staff.ustc.edu.cn/~network/mmt/JPEG_sample_code20220108.rar) | 媒体编解码 | 设计型 |
| 实验十 | [mp3音乐文件播放](http://staff.ustc.edu.cn/~network/mmt/mm_mp3.pdf) | [下载](http://staff.ustc.edu.cn/~network/mmt/mm_mp3_eg.zip) | 媒体编解码 | 设计型 |
| 实验十一 | [动图像传输](http://staff.ustc.edu.cn/~network/mmt/mm_net_movie.pdf) | 下载 | 多媒体通信 | 设计型 |
| 实验十二 | [多媒体电子邮件传输](http://staff.ustc.edu.cn/~network/mmt/mm_mail.pdf) | [下载](http://staff.ustc.edu.cn/~network/mmt/mail.zip) | 多媒体通信 | 设计型 |
| 实验十三 | [IP电话简易模型](http://staff.ustc.edu.cn/~network/mmt/mm_net_voice.pdf) | 下载 | 多媒体通信 | 设计型 |
| 实验十四 | [交互式视频点播](http://staff.ustc.edu.cn/~network/mmt/mm_vod.pdf) | 下载 | 多媒体通信 | 演示型 |
| 实验十五 | [RTP/RTCP协议基础](http://staff.ustc.edu.cn/~network/mmt/mm_rtp.pdf) | 下载 | 多媒体通信 | 设计型 |
| 实验十六 | [Android下的音频编程](http://staff.ustc.edu.cn/~network/mmt/exp_android_audio.pdf) | [下载](http://staff.ustc.edu.cn/~network/mmt/AudioSample.rar) | 手机多媒体 | 设计型 |
| 实验十七 | [Android下的摄像头编程](http://staff.ustc.edu.cn/~network/mmt/exp_android_video.pdf) | [下载](http://staff.ustc.edu.cn/~network/mmt/VideoCapture.rar) | 手机多媒体 | 设计型 |
| 实验十八 | [Android下使用Tensorflow模型](http://staff.ustc.edu.cn/~network/mmt/exp_android_tensorflow.pdf) | [下载](http://staff.ustc.edu.cn/~network/mmt/android_tensorflow.rar) | 手机多媒体 | 设计型 |
| 实验十九 | [SDL+ffmpeg播放mp4](http://staff.ustc.edu.cn/~network/mmt/exp_sdl_ffmpeg.pdf) | [下载](http://staff.ustc.edu.cn/~network/mmt/SDL_ffmpeg_player.rar) | 视频解码 | 设计型 |

1. 在高版本的Android系统中，每个 Android 应用都在访问权受限的沙盒中运行。对麦克风的访问权限，除了在AndroidManifest.xml 文件中申明权限外，需要运行时请求权限（使用 RequestPermission）。 [↑](#footnote-ref-1)
2. 在高版本的Android系统中，每个 Android 应用都在访问权受限的沙盒中运行。对照相机的访问权限，除了在AndroidManifest.xml 文件中申明权限外，需要运行时请求权限（使用 RequestPermission）。 [↑](#footnote-ref-2)
3. 当项目的字符集为Unicode，而编程时没对ASCII字符串做转换，编译时会提示类似如下错误：

   E0167 "const char \*" 类型的实参与 "LPCTSTR" 类型的形参不兼容

   C2664 “void CWnd::SetWindowTextW(LPCTSTR)”: 无法将参数 1 从“const char [10]”转换为“LPCTSTR” [↑](#footnote-ref-3)
4. Kotlin是一个用于现代多平台应用的静态编程语言，由 JetBrains 开发。Kotlin可以编译成Java字节码，也可以编译成JavaScript，方便在没有JVM的设备上运行。除此之外Kotlin还可以编译成二进制代码直接运行在机器上（例如嵌入式设备或 iOS）。Kotlin的设计目标是创建一种兼容Java的语言，让它比Java更安全，能够静态检测常见的陷阱；让它比Java更简洁。Kotlin v1.0于2016年2月15日发布。在Google I/O 2017中，Google宣布在Android上为Kotlin提供一等支持。 [↑](#footnote-ref-4)
5. 集成开发环境（IDE，Integrated Development Environment ）是用于提供程序开发环境的应用程序，一般包括代码编辑器、编译器、调试器和图形用户界面等工具。集成了代码编写功能、分析功能、编译功能、调试功能等一体化的开发软件服务套件。 [↑](#footnote-ref-5)
6. DSL（domain specific language），领域专用语言：专门解决某一特定问题的计算机语言，例如数据库查询语言SQL、和正则表达式。 [↑](#footnote-ref-6)
7. Kotlin是一种面向JVM的新语言。Kotlin的设计目标是创建一种兼容Java的语言，让它比Java更安全，能够静态检测常见的陷阱；让它比Java更简洁。 [↑](#footnote-ref-7)
8. 实测在“Android Studio Koala Feature Drop | 2024.1.2”版本下使用Groove DSL自动构建的Project编译时会出现较多错误，故不推荐使用Groove DSL。 [↑](#footnote-ref-8)
9. XML即Extentsible Markup Language(可扩展标记语言)，可以使用XML文件来描述UI的效果。 [↑](#footnote-ref-9)