Open Source Formats: PDF, HTML, and EPUB

Let's start with the content publishing formats that support digital audio and that have been defined by industry groups, such as **EPUB** and **HTML**, or that have been "open sourced," such as the Adobe **Portable Document Format**, or **PDF**. These formats support a wide range of digital audio file formats. I'm starting with the open formats because they are usable across every type of hardware device covered in this chapter. I'll start with platforms with the widest support.

Portable Document Format: Digital Audio in PDF

The Adobe PDF format is utilized by the Adobe Acrobat Reader, which is used around the world for publishing rich media documents that can include digital audio, digital video, and i3D (interactive 3D). Acrobat Reader is free and the PDF format is open source, but the Adobe Acrobat Professional series of publishing tools are paid software packages—and worth the money if you need to publish via this widely accepted publishing format.

The PDF format supports two digital audio formats, MP3 and MPEG-4 AAC LC; the best option is MPEG-4 AAC LC, as far as the quality-to–file size ratio is concerned. This is because the audio encoding algorithm is more recent, and therefore, much more advanced mathematically. As you saw in Chapter 12, MPEG4 AAC LC gets you far better data footprint results than MP3, so use MPEG4 AAC LC encoding whenever possible for the digital audio assets in your projects.

It used to be that a PDF was only used for creating business documents. However, it has been adopted for e-book formats; in fact, you might be reading my books using the PDF format. Other e-book formats include Kindle (MOBI) and EPUB (EPUB3), which are also covered in this chapter.

Another advantage of the PDF format is that it offers **digital rights management** (DRM) support. This allows you to copy-protect (lock) your document if you want to sell it. Adobe has a PDF Server product that incorporates this DRM feature and allows you to better market PDF content.

It's rumored that the other two publishing formats covered in this section of the chapter are also looking at adding DRM support in the future. Let's look at HTML5 next.

HyperText Markup Language: HTML5 Digital Audio

You already know how to use the `<audio>` tag from Chapter 13; thus, you know that HTML5 supports three digital audio formats—MP3, Ogg Vorbis, and PCM WAVE. The best is Ogg Vorbis, as far as the quality-to–file size ratio is concerned. As you saw in Chapter 12, Ogg Vorbis rivals MPEG4 AAC. This is quite impressive for an open format.

It used to be that HTML5 was only used for creating web site designs, until web browser manufacturers decided to utilize their browser code to create HTML5 operating systems for consumer electronics devices, given the success of Android, Bada, and iOS.

Putting this browser code, with **app launch icon** support, on top of the Linux kernel, produced the Chrome OS (Motorola), Firefox OS (Panasonic iTV), and Opera OS (Sony Bravia iTVs).

There's also the Tizen OS (Samsung), which is based on a Linux OS created by Linus Torvald. It also uses HTML5. Tizen OS is managed by The Linux Foundation. HTML5 is easy to implement, thanks to an open source WebKit API that is also available in Android Studio 1.4 (Android OS 5.4).

HTML5 application and web site publishing is an excellent way to deliver content across all embedded mobile OS, desktop, and web browser platforms. This is why DRM is the future of HTML5 and why I showed you how to implement digital audio assets by using the **<audio control>** method.

Next, let's take a closer look at the open source EPUB 3 publishing standard, used for e-books, and soon for much more.

Electronic Publishing: Digital Audio in EPUB3

The EPUB specification is a distribution and interchange format standard for digital publications and documents. EPUB 3, the third major release of the open EPUB standard, consists of four specifications, each defining an important component of an overall EPUB document. **EPUB Publications 3** defines publication-level semantics, as well as conformance requirements for EPUB 3 documents. **EPUB Content Documents 3** defines XHTML, SVG, and CSS3 profiles for use in the context of your EPUB 3.0 publications. **EPUB Open Container Format 3.0**, or OCF3, defines a file format and is a processing model for encapsulating sets of related resource assets in one ZIP file format (EPUB Open Container). **EPUB Media Overlays 3.0** defines a format and a processing model for the data synchronization of text with digital audio assets.

EPUB 3.0 has been widely adopted as a format for digital books, also popularly known as **e-books**. The 3.0 specification significantly increased the EPUB format's capability, so it is capable of supporting a range of new media publication requirements. These include complex layout, new media and interactivity, and international typography (fonts) support.

The hope is that EPUB 3 will be utilized for a broad range of content, including books, magazines, and educational, professional, and scientific publications.

EPUB 3 supports audio (and video) embedded in documents by using the HTML5 `<audio>` tag (and `<video>` tag) elements. They inherit the same functions and feature set that these tags provide in HTML5. EPUB 3 supports MP3 audio as well as MPEG-4 AAC LC in an `.m4a` container (extension).

Another impressive new media feature in EPUB 3 is called **Media Overlay Documents**. If pre-recorded narration is available for your multimedia publication, such as what you've created in this book, these Media Overlay Documents offer the ability to synchronize your digital audio samples with the text inside the publishing content document (EPUB3 publishing platforms).

Open Platforms: Java, Android, and Kindle

The next set of formats that I am going to cover are open source and free for commercial use, but do not run across all hardware devices and are not industry specific, but instead are owned by major industry hardware and software manufacturers. Oracle owns Java and JavaFX; Google owns Android; and Amazon owns Kindle (MOBI) and Kindle Fire,

which uses the KF8 format. Let's cover these based on the genres or types of consumer electronics devices that run on these formats, starting with e-book readers, since the three formats just covered are all widely used for delivering e-books as well (as you can see on the Apress web site when you purchase their titles).

e-Book Readers: Kindle Fire, Android, Java, and PDF

The e-book reader hardware device is actually an Android tablet, which is why I included Android in the header of this section. The world's most popular e-book reader, the Kindle Fire, runs on Android OS, as does the Sony e-book reader, and the Barnes & Noble NOOK e-book reader. Even the Apple iPad runs Kindle, EPUB3 and PDF e-book titles; so do Blackberry tablets and Microsoft Surface tablets. The reason that I included Java in the headers is because Kindle has Java capabilities for interactive e-books, and Android uses Java. Since e-book readers also read PDF files, I included PDF in the header.

Because most e-book readers are actually Android tablets or iPads, there are a large number of platforms; you saw the key open ones in Chapter 13 to develop digital audio content with.

This means that you can deliver a digital audio content and user experience through an Android application, an HTML5 application, a JavaFX application, an HTML5 web site, a Kindle 8 e-book, an EPUB3 e-book, a NOOK e-book, or an interactive new media PDF document. This gives you a ton of flexibility for publishing audio to e-book readers.

Since this is all done using Java, JavaFX, Android Studio, and the HTML5 `<audio>` tag, the basics of how this is accomplished was covered in Chapter 13.

iTVs: Android TV, Java, JavaScript, and HTML5

The iTV, or interactive television set, is the most recent consumer electronics device to hit the marketplace. iTV devices are expected to explode in sales in 2016 and 2017. This is the reason that Google has developed a specialized version of Android SDK (software development kit) for iTVs called the **Android TV API** (application programming interface).

There are **HTML5 OS iTV** products available from Samsung (Tizen OS), Panasonic (Firefox OS), and Sony (Opera OS). So the iTV consumer electronic device is much like an e-book reader in that it allows you to create and deliver digital audio content by using Java or JavaFX (Android OS or HTML5 OS), HTML5 markup, CSS3, and JavaScript (iOS, Android OS, HTML5 OS).

An important feature that iTVs will have is high-quality digital audio reproduction. Manufacturers realize their iTV products will be installed in home theaters and living rooms, where elaborate sound systems are likely attached to the iTV using **SPDIF** digital audio or **stereo RCA** ports located on the rear of the iTV device.

It's also important to realize that with iTVs, your viewers will pay closer attention to content streams, both audio and video. This is not always the case on devices such as smartphones and automobile dashboards.

If you want to deliver digital audio content across all the iTV platforms, you should use HTML5. Android and iOS support HTML5, but HTML5 OS and web sites do not support Android and iOS applications. The other side of the decision is that Apple and

Google Play have more advanced app stores, so if you are going to monetize your digital audio content, you should consider developing apps with Java (Android) or JavaFX (iOS) more than using JavaScript under HTML5 operating systems and HTML5 browsers.

Smartwatch: Android WEAR, Java, and HTML5

The **smartwatch** is another consumer electronics device that recently hit the market. Smartwatch devices are also expected to explode in sales in 2016 and 2017, primarily because there are hundreds of manufacturers manufacturing them. The densely populated watch industry is moving to release smartwatch products so that they do not lose market share to consumer electronics manufacturers, such as LGE, Sony, Motorola, and Samsung, who already have several smartwatch products. One of the first custom Android APIs that Google developed was the **Android WEAR** with its **Watch Faces API**.

Digital audio is a very important feature that the smartwatch devices support, because 16-bit, 48 kHz audio is built into the hardware, and more importantly, because there is a massive high-end Bluetooth audio headphones market.

What this means is that the smartwatch product is like a MP3 player for your wrist: it doesn't require much screen-display real estate and it can provide a professional audio playback results. This is significant for digital audio producers and digital audio application developers.

Another important feature of smartwatches is that you'll be able to combine your digital audio assets with other highly functional attributes, such as time, date, weather, fashion, and health (especially fitness and physical health monitoring features like heart and pulse rate) hardware input.

Once smartwatch screen resolutions increase from 320 pixels to 480, 640, or 800 pixels, even more functionality becomes available to developers. The Huawei smartwatch already features a 400×400–pixel screen. So, high-resolution smartwatches should appear in 2016 or 2017, given that smartphones have 4K screens that are only 5 to 7 inches. An 800-pixel smartwatch screen is certainly possible, because the technology already exists.

As long as your smartwatch users have quality, Bluetooth headphones, and as long as you process and optimize your 16-bit 48 kHz digital audio assets perfectly, and use the lossless FLAC codec or high-quality settings for the Ogg Vorbis or MPEG-4 AAC codecs, you should rock the socks off your customers!

Auto Dashboard: Android AUTO, Java, and HTML5

The **automobile dashboard** is another recent consumer electronics device to hit the mass market. Auto dashboard devices are expected to become standard in cars by 2016 or 2017. A number of manufacturers already have them as standard equipment and all of the automobile manufacturers have signed on with Google to support Android AUTO, the custom Android SDK for automobile dashboard applications. Automobiles are a hyper-completive industry not likely to be left behind as far as technology is concerned, so it is a logical market for Android and HTML5 operating systems to get into.

Digital audio is a very important feature for auto dashboards to support, because extensive (and expensive) ultra-high quality audio playback hardware is often built right into the body of an automobile, especially in more expensive brands, which are almost half of the automobile brands on the market.

Digital audio is also the best fit for Android AUTO apps, because there are stringent guidelines regarding tasking driver attention off the road. The digital audio Android AUTO apps do not require the user to look at any display screen, and are thus the safest type and should pass muster in the Google Play Automotive App section of the online store.

There are entirely new consumer electronic device types that thus far have very few apps, especially digital audio apps, so the opportunity for audio developers is nothing short of immense. So make some big money!

Smartphones and Tablets: Android, Java, and HTML5

Smartphones or **tablets** have been around a while, as has the hybrid between the two, commonly referred to as a **phablet**. The Android OS covers all of these device types, as well as personal computers that run the Android OS. There are billions of smartphones and tablets, and nearly a hundred consumer electronics manufacturers making products for the open source Android platform. For this reason, it is an amazing opportunity for audio content and applications, as there are not as many digital audio-centric applications as there are video-centric or imaging (visually oriented) applications.

Since smartphones and tables have been on the market for a while, they are most likely to upgrade their 16-bit 48 kHz CD audio hardware support to 24-bit HD digital audio hardware. And since Android supports the 24-bit FLAC codec, pristine digital sample audio is within the reach of digital audio editors and compositors.

iTVs are also likely to include 24-bit HD Audio support, as customers will demand HD Audio and SPDIF connectors for their UHD home theater installations. In fact, there are already UHD iTV products. I would be quite surprised if these did not already feature HD Audio hardware. Again, a bright future awaits audio editors, as CD- and HD-quality digital audio apps are still rare in the market.

Game Consoles: Android, Java, JavaFX, and HTML5

Since Android, Java, JavaFX, and HTML5 now support **OpenGL ES 3.1**, a plethora of advanced game console products have appeared, affordably priced between $50 and $100. This is yet another opportunity waiting for digital audio editing gurus, which you'll soon be, once you practice what you've learned in this book. These consoles run on Android and therefore support Java and HTML5, as well as JavaFX apps or Android applications, and even e-books, for that matter. There are more than a dozen brands available now.

Some major industry brands (manufacturers) are producing game controllers with Android computers inside, for instance an NVIDIA SHIELD or GameStick. Other major manufacturers, such as Amazon, manufacture a game-console iTV hybrid product, such as the Amazon Fire TV. Others, such as OUYA and GamePop, make STB (set-top box) products that game controllers (and iTVs) plug into. Some, such as OUYA and Razer ForgeTV, come with both the STB and the game controller, for a complete gaming package.

Since all of these game controllers support Android, you can utilize the audio formats covered in this book, and if you use HTML5 or EPUB3, you can use Ogg Vorbis, MP3, or MPEG4 AAC—all of which provide good results. (I covered the code for doing this in Chapter 13.)

Future Devices: Robots, VR, and Home Appliances

The future of Android SDKs will surely bring more custom APIs. I expect to see **Android VR** for virtual reality goggles, as well as **Android HOME** for home appliances or home control units, and maybe even an **Android ROBOT** SDK for Android-based robots. I have already seen many of these products in the marketplace for some time, so it's up to Google to provide custom APIs for these product genres, all of which will be great audio app platforms for digital audio editors, multimedia producers, and developers.

Audio will be an important component for all of these devices. I expect at least two of these genres, home appliances and virtual reality, to support HD audio to increase the user experience level (virtual reality), and because home theaters use HD Audio (home appliances and control units). Robots will probably stick with 16-bit 48 kHz audio, as that is enough bandwidth to produce professional results. So if you follow the work processes outlined in this book, you should be able to produce pristine results, especially if you use FLAC codec at a 16-bit sample resolution and a 48 kHz sampling rate.

Paid Software Platforms: iOS or Windows

The last section of this chapter covers formats that are not open source; that is, they involve paid software. Paid hardware is required to develop Apple platforms. Some formats require the company that owns the platform to approve your app before it can be sold in the app store. It is important to note that you are able to get around this approval process by developing with HTML5 for these platforms, or by using JavaFX; therefore, you can still deliver content for your clients without having to invest thousands of dollars in hardware (for iOS) and software (Windows Visual C++ or C# software development packages).

Apple iPhone and iPad: Supported Audio Formats

The Apple iPhone and iPad run iOS and support the following audio formats, many of which are covered in this book: MPEG-4 AAC (16 to 320 Kbps), AIFF, AAC Protected (MPEG-4 DRM format in the iTunes Store), MP3 (CBR 16 to 320 Kbps), MP3 VBR, Windows WAV PCM, and the proprietary Apple Lossless and Audible formats (V2 through V4). Apple QuickTime is also proprietary, and so the MPEG-4 and WebM video formats are far more widespread in use for this reason.

Other closed or paid formats, like Flash and PowerPoint, have suffered the same fate: replaced with open, free-for-commercial use formats such as Java, HTML5, and EPUB3.This is why I have focused primarily on the open sourced digital audio codecs and formats, or those like MPEG-4, which are licensed for use in JavaFX, Android Studio, and HTML5.

Windows Phone: Supported Digital Audio Formats

The Windows Phone 8 supports the following audio formats (covered in the book): MPEG-3, WMA Standard 9.2, WMA Pro, MPEG-4 AMR-NB, and MPEG-4 AAC LC. The Windows 7, 8.1 and 10 operating systems also support these formats. If you are developing for your corporate clients running Windows 10, use the WMA Pro codec, as it probably has the best codec algorithms and HD Audio support.

Summary

In this final chapter, you looked at the digital audio publishing concepts, principles, and file formats used to compress and decompress digital audio assets, as well as to publish and distribute them to your end users. You also looked at many of the different formats, platforms, and devices available to you for developing digital audio content.

I hope that you have enjoyed this journey through digital audio editing, compositing, programming, and publishing concepts and work processes, and that you now have fundamental knowledge of digital audio, which you can build on in your new media design, multimedia development, and content publishing endeavors. Keep an eye out for my other books, covering Android Studio, Java and JavaFX, HTML5, and other new media genres, such as digital illustration and digital image compositing.

Index

A

Android PacKage (APK), 126
Adaptive Multi-Rate (AMR), 31
Algorithmic effects, 56
    Amplify effect, 56
    Delay effect, 61
    Equalization effect, 59
    Low Pass Filter effect, 62
    Notch Filter effect, 63
    Pitch Shifting effect, 57
    Reverb effect, 60
    Speed effect, 58
Aligning tracks
    Align selection, 88–89
    Envelope Tool, 91
    multiple tracks, 88
    text field, 89–90
Align tracks feature, 88
Amplify effect, 56
Analog audio
    amplitude, 7–8
    resilient membranes, 6
    wave cycle, 7
Android
    home appliances, 137
    ROBOT, 137
    VR, 137
Android studio
    AudioTrack class, 127–128
    MediaPlayer class, 128–129
    MediaRecorder class, 129–130
    SoundPool Class, 126–127
Apple iPhone and iPad, 137
Audacity
    audio transport controls, 24
    menu sequence, 25
    monitoring, 23
    record button, 24
Audacity Analyze menu, 65
    Regular Interval Labels, 65
    Sample Data Export, 67
Audacity software
    audacityteam.org, 1
    audio file formats, 4
    default installation options, 3
    Finish button, 4
    GNU licensing information, 3
    installation, 4
    LAME 3.99.3 and FFMPEG 2.2.2 libraries, 6
    language selection, 2
    operating system link, 5
    plug-ins and sound libraries, 4
Audacity synthesizer
    Fire Explosion generator, 100–101
    harmonic noise, 103–104
    KLSTRBAS generator, 101–102
    Oxygene Surf generator, 96
    Pluck generator, 104–105
    Track generator, 98
    Tuning Fork generator, 99–100
    virtual surf waveforms, 97
Audio buffer, 128
Audio Coding AAC-LC, AAC-ELD/HE-AAC, 31
Audio optimization
    AIFF file format, 109
    baseline PCM file, 108–110
    device compatibility, 107
    Edit Metadata dialog, 110
    FLAC audio format, 111
    MPEG3 audio format, 112–113
    MPEG4 AMR format, 114–115

W. Jackson, *Digital Audio Editing Fundamentals*, DOI 10.1007/978-1-4842-1648-4