Application Development Using NexStreaming
NexPlayer* SDK

Lana N. Lindberg, Software Applications Engineer

Introduction

NexStreaming is a global mobile software company with headquarters in Seoul, Korea and branches in Spain, United States, Japan, and China. Its most popular product, NexPlayer* SDK, is a player SDK used by some of the most famous video service providers integrated into mobile applications. The player is compatible with all the leading DRM technologies in the industry. Moreover, it can be combined with other complementing technologies such as advertisement insertion, audience measurements, or audio enhancements. NexStreaming NexPlayer SDK provides audio and video decoding and playback services. Application developers can use the SDK to build custom efficient multimedia players quickly. NexPlayer SDK is reliable and robust and has proven to be compatible with international standards. This paper describes how to create an x86 player app using the NexPlayer SDK.

Platform Compatibility

The NexPlayer SDK is optimized for x86, so it is fully supported by x86 devices. NexPlayer SDK supports the following:

- Android* 1.6 or above
- mp4, 3gp, avi, asf, piff video file formats
- HTTP Live Streaming version 5.0, 3GPP Progressive Download, AES128, HTTPS protocols h.264, AAC, AAC+, eAAC+ codecs. Both software and hardware codecs are supported by the SDK
- .smi, .srt, .sub, 3GPP timed text, TTML closed captions (PIFF/CFF only), CEA 608 and CEA 708 closed captions, and Web Video Text Tracks (WebVTT)
How to create an x86 player app using NexPlayer SDK

You need to request the SDK from NexStreaming at [http://www.nexstreaming.com/downloads-sdk](http://www.nexstreaming.com/downloads-sdk). Once you download the SDK and the demo app, the optimization for the Intel® chipsets will be automatically included. Full documentation is included with the SDK. Refer to the SDK documentation and samples to select the APIs appropriate for your application. App development using the SDK is very easy and straightforward. You can use sample code as a guide. It takes about one hour to develop a complete app with NexPlayer SDK.

To integrate NexPlayer SDK into an x86 Android app and ensure the best experience with the NexPlayer SDK on Intel® devices, you need to perform the following simple steps.

- Copy the libs contained in the SDK/libs/ folder into the assets/x86 folder of your project.
- Copy the libs contained in the SDK/libs folder into the libs/x86 folder of your project.
- Copy the source files contained in the SDK/src folder into the src/com/nexstreaming/nexplayerengine folder of your project.

The NexPlayer SDK will detect that change and use those libraries to take advantage of the Intel resources. Once all libraries are in these directories, switch between ARM and x86 version of an app is automatic and handled by the SDK. If you want to integrate a newer version of NexPlayer SDK, the only thing you need to do is to overwrite the library files above.

NexPlayer SDK includes a large number of libraries including DRM libraries. These libraries are in the app/assets/x86 directory. Required libraries consist of engine, decoders, and the rendering layer:

- libnexplayerengine.so
- libnexalfactory.so
- libnexadaptation_layer_for_dlsdk.so
- libnexralbody_audio.so
- libnexralbody_video_opengl.so
- libnexral_nw_ics.so
- libnexral_nw_jb.so
- libnexcal_oc_ics.so
- libnexcal_oc_jb.so
- libnexcralbody_mc_jb.so
- libnexcal_in_aac_x86.so
- libnexcal_in_mp3_x86.so
- libnexcal_in_amr_x86.so

Some library names contain abbreviations “ics” for Ice Cream Sandwich and “jb” for Jelly Bean. If your app only supports certain version(s) of Android, you can remove the libraries for the unsupported version(s) of the OS.

The libraries that provide support for codecs are:

- libnexcal_h264_x86.so - video lib for H.264
- libnexcal_aac_x86.so - audio lib for AAC, AAC-Plus, and HE-AAC
The following libraries support text captions:

- libnexcal_mp3_x86.so - for MP2 and MP3
- libnexcal_3gpp_x86.so - for 3GPP timed text captions
- libnexcal_closedcaption_x86.so - for CEA 608 and CEA 708 closed captions
- libnexcal_ttml_x86.so - for TTML (CFF) timed text captions
- libnexcal_webvtt_x86.so - for WebVTT text tracks

To reduce the size of the app, include only the libraries that you need for your app.

The libraries from app/libs/x86 need to be loaded in InitManager() in Java* source code for the application file. For example, for “NexHDSample” add the corresponding x86 library into app/src/NexHDManager.java in initManager():

```java
System.loadLibrary("NexHTTPDownloaderSample_jni");
```

**How to display your x86 app’s video on the screen using the SDK**

There are two ways to display a video using NexPlayer SDK: NexVideoRenderer and the OpenGL* renderer. NexVideoRenderer is the recommended way to display video. NexVideoRenderer abstracts the complexities of surface handling and video rendering tasks by choosing the most appropriate renderer based on the device and version of the OS. To implement NexVideoRenderer, do the following:

1. Pass in a context (android.content.Context) to the constructor.
2. Set up listeners (NexPlayer.IListener and NexPlayer.IVideoRendererListener).
3. Create an instance of NexPlayer.
4. Perform the necessary setup for NexPlayer (NexPlayer.setNexALFactory and NexPlayer.init).
5. Call init with the NexPlayer instance (NexVideoRenderer.init).
6. Add the NexVideoRenderer instance as a view to your layout.

A complete example of this renderer is in
NexPlayerSample/src/com.nexstreaming.nexplayerengine/NexVideoRenderer.java.

**Live Streaming**

HTTP Live Streaming supports multiple audio and video streams. setMediaStream() API allows these streams to be selected from the user interface while content is being played. This is supported by the SDK and is mentioned in this paper just for your information. Three possible use cases are available:

1. A variant playlist with alternative audio. In this case video and audio can be selected independently.
2. A variant playlist with alternative video. Here each track contains both audio and video, but alternative video streams are available (for example, different camera angles or views of the same content).
3. A combination of a variant playlist with alternative video and audio. This use case is a combination of the above cases, where a main video stream provides video tracks at different bitrates but INCLUDES the same audio, and separate audio tracks are available for optional language selection.

**PnP Analysis of x86 NexPlayerDemoApp**

In the following analysis I evaluated launch and idle as well as playback of local mp4 files and sample streams of a sample app (“NexPlayerDemoApp”) developed using NexPlayer SDK. The analysis was performed with the VTune™ analyzer for Android and Intel® SoC Watch on an Intel® Atom™ processor-based tablet Z3740 @1.6 GHz with Intel® HD Graphics (Gen 7) and Wi-Fi* connected, on Android 4.4.2 (KitKat).

<table>
<thead>
<tr>
<th>Intel® Atom™ processor-based tablet Z3740 Baseline</th>
<th>Package</th>
<th>% Spent in State for core0</th>
<th>% Spent in State for core1</th>
<th>% Spent in State for core2</th>
<th>% Spent in State for core3</th>
<th>% Spent in GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C0</td>
<td>4.00</td>
<td>2.50</td>
<td>.40</td>
<td>.40</td>
<td>.3</td>
</tr>
<tr>
<td></td>
<td>C6</td>
<td>95.00</td>
<td>95.10</td>
<td>98.30</td>
<td>98.10</td>
<td>99.5</td>
</tr>
</tbody>
</table>

**NexPlayerDemoApp at Idle**

<table>
<thead>
<tr>
<th>Package</th>
<th>% Spent in State for core0</th>
<th>% Spent in State for core1</th>
<th>% Spent in State for core2</th>
<th>% spent in GPU</th>
<th>% spent in GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>4.8</td>
<td>3.7</td>
<td>1.2</td>
<td>0.9</td>
<td>0</td>
</tr>
<tr>
<td>C6</td>
<td>93.5</td>
<td>94.1</td>
<td>93.4</td>
<td>93.9</td>
<td>100</td>
</tr>
</tbody>
</table>

High C0 times on the Intel® Atom™ processor-based tablet Z3740 during the application idle will result in increased power consumption, but this is not the case with NexPlayerDemoApp. The numbers are very close to Z3740 baseline.

**NexPlayerDemoApp at Launch**

<table>
<thead>
<tr>
<th>Package</th>
<th>% Spent in State for core0</th>
<th>% Spent in State for core1</th>
<th>% Spent in State for core2</th>
<th>% spent in GPU</th>
<th>% spent in GPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>4.8</td>
<td>3.7</td>
<td>1.2</td>
<td>0.9</td>
<td>0.9</td>
</tr>
<tr>
<td>C6</td>
<td>93.5</td>
<td>94.1</td>
<td>93.4</td>
<td>93.9</td>
<td>99.9</td>
</tr>
</tbody>
</table>

The numbers are very close to Z3740 baseline.
Video Playback
During video playback, the x86 version of NexPlayerDemoApp utilizes an average of 33% of the CPU. The activity does not show abnormalities and is consistent with the core total.

Figure 2. Video Playback Time View for x86 NexPlayerDemoApp.

Live Streaming
During live streaming, the x86 version of NexPlayerDemoApp utilizes an average of 25% of the CPU. The activity does not show abnormalities and is consistent with the core total.

Figure 3. Live Streaming Time View for x86 NexPlayerDemoApp.
Battery Consumption

x86 NexPlayerDemoApp utilizes ~3.8W of power during the combination of live streaming (taking the majority of time), video playback, and a very short idle in x86 version.

Conclusion

Developing apps with the NexPlayer SDK is fast and easy and proves to be efficient with low power utilization on x86 mobile devices.

About the Author

Lana Lindberg is a member of the Intel® Software and Solutions Group (SSG), Developer Relations Division, Intel® Atom™ Processor High Touch Software Enabling team. Before joining SSG, Lana was OpenGL ES test developer in graphics software validation team for the Ultra Mobility Group.

References and Helpful Links

2. NexStreaming website www.nexstreaming.com
Notices

INFORMATION IN THIS DOCUMENT IS PROVIDED IN CONNECTION WITH INTEL PRODUCTS. NO LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE, TO ANY INTELLECTUAL PROPERTY RIGHTS IS GRANTED BY THIS DOCUMENT. EXCEPT AS PROVIDED IN INTEL'S TERMS AND CONDITIONS OF SALE FOR SUCH PRODUCTS, INTEL ASSUMES NO LIABILITY WHATSOEVER AND INTEL DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY, RELATING TO SALE AND/OR USE OF INTEL PRODUCTS INCLUDING LIABILITY OR WARRANTIES RELATING TO FITNESS FOR A PARTICULAR PURPOSE, MERCHANTABILITY, OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS OTHERWISE AGREED IN WRITING BY INTEL, THE INTEL PRODUCTS ARE NOT DESIGNED NOR INTENDED FOR ANY APPLICATION IN WHICH THE FAILURE OF THE INTEL PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR.

Intel may make changes to specifications and product descriptions at any time, without notice. Designers must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Intel reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The information here is subject to change without notice. Do not finalize a design with this information.

The products described in this document may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an order number and are referenced in this document, or other Intel literature, may be obtained by calling 1-800-548-4725, or go to: http://www.intel.com/design/literature.htm

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark* and MobileMark*, are measured using specific computer systems, components, software, operations, and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products.

Any software source code reprinted in this document is furnished under a software license and may only be used or copied in accordance with the terms of that license.

Intel, the Intel logo, Intel Atom, and VTune are trademarks of Intel Corporation in the US and/or other countries.

Copyright © 2015 Intel Corporation. All rights reserved.

*Other names and brands may be claimed as the property of others.