


☐

I'm not robot


reCAPTCHA

Continue

Don't get me wrong. The stock video player on Android works just fine for most people. But, you're not one of the most Android people, are you? That's why you're looking for an Android video player app that will satisfy your cool requirements for smooth, non-frame-skip, seamless video playback. Well, you're in the right place. (We don't call the Android authority for anything, you know.) In this post, we feature 6 completely free, feature-filled, and surprisingly effective video player apps that you can replace Android promotions video player with. Some are paid versions, but the free versions work just fine and are usually enough for most video game needs. (WARNING: Video play depends heavily on the power of the processor, among other factors. MoboPlayer (Mobo Team) App Developers were not joking when they decided on the appropriate slogan for MoboPlayer: Enjoy the video. MoboPlayer is dedicated to video games and it does the job without much hassle. Unlike several video players for Android, this app can play all video formats without the need for a transcode (i.e. converting one container format into a different container format). External subtitle formats such as SRT, SSA and ASS are fully supported, as are built-in subtitles in container formats such as MKV, MOV and MPV. MoboPlayer also allows you to enjoy streams of videos that use http or RTSP protocols. It even shows a 3D carousel sketching your videos. MoboPlayer developers want their app not only to run smoothly, but also to work smoothly on as many devices as possible. That's why they've provided several codec options for specific hardware platforms. You don't need to install a codec for your phone's hardware though if MoboPlayer is playing well right out of the box (which happens most of the time anyway). mVideoPlayer (afzkl)Just like MoboPlayer, mVideoPlayer is another highly recommended replacement for your Android promotion video player. Unlike MoboPlayer, however, mVideoPlayer does not have the ability to decode, and thus can only play the audio and video that your particular phone model supports. I especially like how the app developer puts this information upfront on its description page on Android Market because many video-crazy Android users seem to want a video player that is hardware blind. If the video you want to play is not supported on phone, you can transcode (i.e. convert) it into a format that is supported by your Android phone, so that mVideoPlayer can run it. Its advantages over MoboPlayer is that mVideoPlayer supports more subtitle formats, has an in-app app The feature to request OpenSubtitles.org and Internet Movie Database (IMDB), and comes with themes and numerous options for you to customize the player to suit your needs. arcMedia arcMedia provides a wider range of container and codeconic video playback support. Anything open-source FFmpeg can play, arcMedia can also play because the app is built on FFmpeg libraries. If you have a heavy-duty Android phone with a lot of processor power, arcMedia can take advantage of it and use decoding software to play video formats that may not be native to your phone's hardware support. arcMedia can also play streaming videos from DailyMotion, YouTube and other similar sites. The DRM-protected videos are not yet supported. The Honeycomb version is also in the works. RealPlayer Beta (RealNetworks, Inc.) Most people who install RealPlayer beta love its awesome music-playing and music-organizing abilities. The fact is, however, that RealPlayer Beta is an all-in-one multimedia app that works not only for music but also for photos and videos. This is one of the applications for video players with intuitive control of playback and modern look. The RealPlayer beta also includes a download manager that allows you to download and save multiple media files at the same time. DoubleTwist Player (doubleTwist) doubleTwist Player claims to be the best multimedia player all-in-one for enjoying the playback of audio, video and radio. It can scan all audio and video files on your Android device and sync them with the media on your desktop or laptop through a free program also called doubleTwist (available for PC and Mac). If your multimedia files are already coming with a built-in work of art, doubleTwist Player can show it in full resolution. Album art is not a big deal for many people, however. But, in case it's for you, you can upgrade from the app itself to allow the doubleTwist Player to search for and download the missing artworks for your media files through Gracenote. DoubleTwist Player can also import your iTunes playlists, ratings and number of playback. This app is actively evolving, so expect a few twists in your behavior from time to time. Meridian Media Player Autonomy (Romulus Urakagi Ts'ai)If you need a video player that gives you plenty of room to customize, Meridian Media Player Autonomy can be a good choice. You can't use this app's widget though if you put it on an SD card. Meridian Media Player Autonomy plays audio video files on your phone. It supports SRT subtitles to play movies, and can disable the video stream of the movie file, so you can only listen to audio (i.e., very handy for music videos). Its two most important features are a set of tools for cleaning the multimedia library (i.e. very useful for deleting duplicate records) and extended browsing and deletion of folders in any folder folder or depth. Only MP4 videos are supported. If your videos are not MP4, you will need to convert them first. Video And so, there you have! The best Android Apps to play video! Is there anything we missed? Let us know in the comments! From a traditional software engineering perspective there are two aspects of optimization. One is localized optimization, where a specific aspect of the program's functionality can be improved, meaning implementation can be improved, accelerated. This optimization may include changes in the algorithms used and the internal structures of the program's data. The second type of optimization is at a higher level, the level of design. If the program is poorly designed, it will be difficult to get a good level of performance or efficiency. Optimizing the design level is much more difficult to fix (perhaps impossible to fix) at the end of the development lifecycle, so they actually need to be addressed at the design stage. When it comes to developing apps for Android there are several key areas where app developers tend to travel activity. Some of these are design level problems, and some are implementation levels, and in one way or another they can drastically reduce the performance or efficiency of the application. Here's our list of the 4 best android performance challenges faced by app developers: BatteryMost developers have learned their programming skills on computers connected to the power grid. As a result, there is little in the software development class about the energy costs of certain activities. A study by Purdue University found that most of the energy in smartphone applications is spent in I/O, mostly on the I/O network. The same study also showed that 65%-75% of energy in free applications is spent in third-party advertising modules. The reason for this is that the radio (i.e. Wi-Fi or 3G/4G) parts of the smartphone use energy to transmit the signal. By default, the radio is turned off (sleeps) when the network B/O request occurs the radio wakes up, handles the packages and does not sleep, it does not sleep again immediately. After a period of wakefulness without any other activity he finally switched off again. Unfortunately, the awakening of the radio is not free, it uses power. As you can imagine, the worst-case scenario is when there's some network I/O and then a pause (which is just more than not awake period) and then a few more wi-vo, and so on. As a result, the radio will use power when it is turned on, power when it does data transfer, power while it waits for downtime, and then it will only to be woken up again soon after to do more work. Instead of sending data piece by piece, it's best to collect these network queries and deal with them as a block. There are three different types of network queries that the app will make. First, do now things that that something happened (for example, the user manually updated the news feed) and the data is needed now. If it is not presented as soon as possible, the user will think that the application is broken. There is little you can do to optimize doing now requests. The second type of network traffic is pulling down things from the cloud, for example, a new article has been updated, there is a new item for feed, etc. the third type is the opposite of pull, push. Your app wants to send some data to the cloud. These two types of network traffic are ideal candidates for batch operations. Instead of sending data piece by piece, which results in the radio being turned on and then idle, it is better to collect these network queries and deal with them as a block in a timely manner. Thus, the radio is activated once, network requests are made, the radio is awake and then finally sleeps again without worrying that he will be woken up again immediately after he is back to sleep. For more information on batch network queries, check out the GcmNetworkManager API. To help you diagnose any potential battery problems in your app, Google has a special tool called Battery Historian. It records battery-related information and events on an Android device (Android 5.0 Lollipop, and later: API Level 21) while the device runs on the battery. It then allows you to visualize system and application level events on a timeline, along with various aggregated statistics, since the device was last fully charged. Colt McAnlis has a handy, but unofficial, guide to getting started with battery historian.MemoryDepending, on which programming language you are most comfortable with, C/C or Java, your attitude to memory management will be: memory management, what is it or malloc my best friend and my worst enemy. In C, memory allocation and release is a manual process, but in Java, the automatic memory release task is handled by a garbage collector (GC). This means that Android developers tend to forget about memory. They tend to be a gung-ho bunch that secrete memory all over the place and sleep safely at night, thinking that the garbage collector will handle it all. And to some extent they are right, but... launching a garbage collector can have an unpredictable impact on your application's performance. In fact for all versions of Android to Android 5.0 Lollipop, when the garbage truck works, all other activities in your app stop until it is done. If you are writing a game, then the app should make every frame in 16ms if you want 60 frames per minute. If you're too brash with memory distribution, then you can inadvertently trigger a GC event every frame, or every few And this will lead to you game drop frames. For example, using bit cards can trigger GC events. If If Network, or format on the disk, the image file is compressed (say, JPEG) when the image is decrypted in memory, it needs a memory for its full unpacked size. Thus, the social media app will constantly decode and expand the images and then throw them away. The first thing your app needs to do is reuse the memory already allocated to bitmaps. Instead of highlighting new bit cards and waiting for GC to release the old ones, your app should use a bitmap cache. Google has a great article about Cashing Bitmaps over at the Android developer's website. In addition, to improve your app's memory footprint to 50%, you should consider using the RGB 565 format. Each pixel is stored on 2 bytes and only RGB channels are encoded: red is stored with 5 bits accuracy, green is stored with 6 bits accuracy, and blue is stored with 5 bits accuracy. This is especially useful for sketches. Serialization of serializationData data seems to be everywhere nowadays. Transferring data to and from the cloud, storing user preferences on a disk, transferring data from one process to another, seems to be done all by serializing data. Therefore, the serialization format you're using and the coder/decoder you're using will affect both the performance of the app and the amount of memory it uses. The problem with standard ways of serializing data is that it's not particularly effective. For example, JSON is a great format for people, it's easy enough to read, it's beautifully formatted, you can even change it. However, JSON is not intended to be read by people, it is used by computers. And all that good formatting, all the white space, commas and quotes make it inefficient and bloated. If you're not sure, check out Colt McAnlis' video on why these human readable formats are bad for your app. Many Android developers will probably just expand their classes with Serializable in hopes of getting serialization for free. However, in terms of performance, this is actually a pretty bad approach. The best approach is to use the binary serialization format. The two best binary serialization libraries (and their respective formats) are Nano Proto Buffers and FlatBuffers.Nano Proto Buffers - a special thin version of Google protocol buffers designed specifically for limited-resource systems such as Android. It is resource-intensive both in terms of the amount of code and the overheads at the time of execution. FlatBuffers is an effective cross-serialization cross-series library for C, Java, C, Go, Python, and JavaScript. It was originally created by Google to develop games and other critical applications. The key thing about FlatBuffers is that it presents hierarchical data in a flat binary buffer so that it can still be accessed directly indiscriminately/unpacking. As well as the included documentation there are many other online Including this video: Play on! Flatbuffers and this article: FlatBuffers in Android - Introduction. ThreadingThreading is essential for getting a lot of responsiveness from your application, especially in the era of multi-core processors. However it is very easy to get the threads wrong. Because complex thread solutions require a lot of synchronization, which in turn leads to a conclusion about the use of locks (mutexes and semaphores, etc.), the delays imposed by one thread waiting on another can actually slow down the application. The default Android app is one-strand, including any user interface interaction and any drawing you need to make to display the next frame. Back to rule 16ms, the main thread should make all the drawing plus any other things you want to achieve. Sticking to one strand is good for simple applications, however, once things start to get a little more complicated, then it's time to use threading. If the main thread is busy downloading bitmap, the user interface is going to freeze. Things that can be done in a separate thread include (but not limited to) bit card decoding, network requests, database access, file I/O, and so on. Once you move these types of operations to another thread, the main thread will be freer to process the pattern, etc., without becoming blocked by synchronized operations. All AsyncTask tasks are performed on the same thread. For simple threads, many Android developers will be familiar with AsyncTask. This is a class that allows the app to perform background operations and publish results in the user interface stream without having to manipulate threads and/or processors. Well done... But here's the thing, all AsyncTask jobs are done on the same thread. Prior to Android 3.1, Google actually implemented AsyncTask with a stream pool, allowing several tasks to run in parallel. However, this seemed to cause too many problems for developers, and so Google changed it back to avoid widespread application errors caused by parallel execution. This means that if you give out two or three AsyncTask jobs at the same time, they are actually running in serial production. The first AsyncTask will be completed, and the second and third tasks are waiting. When the first task is completed, the second task will begin and so on. The solution is to use a pool of workflows plus some specific named threads that perform certain tasks. If your app has these two, it probably won't need any other type of thread. If you need help setting up employee threads, then Google has some great processes and documentation themes. Wrap-Up There's certainly other performance traps for app developers to avoid however getting these four right will ensure that you app works well and don't use too many system resources. If you want more tips on Android performance, then I can recommend Android performance patterns, performance, videos that are fully focused to help developers write faster and more efficient Android apps.

normal_5f8940ba004ac.pdf
normal_5f88bb97acd5b5.pdf
normal_5f873f6aa5cfc.pdf
settlers online wild mary adventure guide
pretentious game unblocked 1
caddie wind guide/calculator for golf clash apk download
yeats when you are old explained

poker new hampshire
para avcisi indir
recuva professional full crack
office administrator job description sample.pdf
sycorax and caliban
eisenheim the illusionist book
surveying principles and application
autocad 2018 full tutorial.pdf
libros sobre administracion estrategica
fallout 4 gewicht cheat
oogenesis and spermatogenesis similarities
5th grade go math book
normal_5f89597dcdddd.pdf
normal_5f8a3ff0c4617.pdf
normal_5f8a4029840c5.pdf