Displaying Bitmaps Efficiently

Learn how to use common techniques to process and load `Bitmap` (/reference/android/graphics/Bitmap.html) objects in a way that keeps your user interface (UI) components responsive and avoids exceeding your application memory limit. If you're not careful, bitmaps can quickly consume your available memory budget leading to an application crash due to the dreaded exception: `java.lang.OutofMemoryError: bitmap size exceeds VM budget`.

There are a number of reasons why loading bitmaps in your Android application is tricky:

- Mobile devices typically have constrained system resources. Android devices can have as little as 16MB of memory available to a single application. The Android Compatibility Definition Document (CDD), *Section 3.7. Virtual Machine Compatibility* gives the required minimum application memory for various screen sizes and densities. Applications should be optimized to perform under this minimum memory limit. However, keep in mind many devices are configured with higher limits.
- Bitmaps take up a lot of memory, especially for rich images like photographs. For example, the camera on the Galaxy Nexus takes photos up to 2592x1936 pixels (5 megapixels). If the bitmap configuration used is `ARGB_8888` (the default from the Android 2.3 onward) then loading this image into memory takes about 19MB of memory (2592*1936*4 bytes), immediately exhausting the per-app limit on some devices.
- Android app UI's frequently require several bitmaps to be loaded at once. Components such as `ListView`, `GridView` and `ViewPager` commonly include multiple bitmaps on-screen at once with many more potentially off-screen ready to show at the flick of a finger.

DEPENDENCIES AND PREREQUISITES

- Android 2.1 (API Level 7) or higher
- Support Library

TRY IT OUT

Download the sample

BitmapFun.zip

VIDEO
DevBytes: Bitmap Allocation

VIDEO
DevBytes: Making Apps Beautiful - Part 4 - Performance Tuning

Lessons

Loading Large Bitmaps Efficiently
This lesson walks you through decoding large bitmaps without exceeding the per application memory limit.

Processing Bitmaps Off the UI Thread
Bitmap processing (resizing, downloading from a remote source, etc.) should never take place on the main UI thread. This lesson walks you through processing bitmaps in a background thread using `AsyncTask` and explains how to handle concurrency issues.

Caching Bitmaps
This lesson walks you through using a memory and disk bitmap cache to improve the responsiveness and fluidity of your UI when loading multiple bitmaps.

Managing Bitmap Memory
This lesson explains how to manage bitmap memory to maximize your app's performance.

Displaying Bitmaps in Your UI
This lesson brings everything together, showing you how to load multiple bitmaps into components like `ViewPager` and `GridView` using a background thread and bitmap cache.