Bug: Android Maps API v2 leaks huge amounts of memory

STATUS UPDATE No update yet. Edit

DESCRIPTION rc...@gmail.com created issue #1

Steps to reproduce:

1. Open the Google Maps API Demos example application provided with SDK. Go to "Basic Map" view for example.
2. Start rotating the phone so that the Activity rotates
3. Watch the Heap size grow.
4. Notice the heap is consumed much faster when using SATELLITES mode, about megabyte per rotation!

****
For developers viewing this issue: please click the 'star' icon to be
notified of future changes, and to let us know how many of you are
interested in seeing it resolved.
****

Links (11)

"http://stackoverflow.com/questions/14523949/google-maps-android-api-v2-supportmapfragment-memory-leak#c… "
"https://developers.google.com/maps/documentation/android/reference/com/google/android/g…"
"https://groups.google.com/group/google-maps-android-… "
" …elease notes for the Feb 2013 update (https://developers.google.com/maps/documentation/android/releases#february_2013) indicate that this bug has been fixed, but it hasn't. The original step
"http://maps.t.al"

See all related links

COMMENTS

cp...@gmail.com <cp...@gmail.com> #2

[Comment deleted]

cp...@gmail.com <cp...@gmail.com> #3

Nobody said nothing about this issue?

kw...@gmail.com <kw...@gmail.com> #4

The dump that I get often is:

```
01-28 16:46:01.284: E/dalvikvm-heap(8157): 1048576-byte external allocation too large for this process.
01-28 16:46:01.284: E/GraphicsJNI(8157): VM won't let us allocate 1048576 bytes
01-28 16:46:01.284: W/System.err(8157): OutOfMemory
01-28 16:46:01.416: D/dalvikvm(8157): JIT code cache reset in 9 ms (1048516 bytes 3/1)
01-28 16:46:01.424: D/dalvikvm(8157): GC_EXPLICIT freed 15518 objects / 718856 bytes in 136ms
01-28 16:46:01.729: E/dalvikvm-heap(8157): 1048576-byte external allocation too large for this process.
01-28 16:46:01.729: E/GraphicsJNI(8157): VM won't let us allocate 1048576 bytes
01-28 16:46:01.807: W/dalvikvm(8157): threadid=28: thread exiting with uncaught exception (group=0x4001d7e0)
01-28 16:46:01.885: E/AndroidRuntime(8157): FATAL EXCEPTION: GLThread 35
01-28 16:46:01.885: E/AndroidRuntime(8157): java.lang.OutOfMemoryError: bitmap size exceeds VM budget
01-28 16:46:01.885: E/AndroidRuntime(8157):     at android.graphics.Bitmap.nativeCreate(Native Method)
01-28 16:46:01.885: E/AndroidRuntime(8157):     at android.graphics.Bitmap.createBitmap(Bitmap.java:468)
01-28 16:46:01.885: E/AndroidRuntime(8157):     at maps.r.h.a(Unknown Source)
01-28 16:46:01.885: E/AndroidRuntime(8157):     at maps.cp.a.a(Unknown Source)
01-28 16:46:01.885: E/AndroidRuntime(8157):     at maps.cp.a.a(Unknown Source)
01-28 16:46:01.885: E/AndroidRuntime(8157):     at maps.cp.a.b(Unknown Source)
01-28 16:46:01.885: E/AndroidRuntime(8157):     at maps.m.n.a(Unknown Source)
01-28 16:46:01.885: E/AndroidRuntime(8157):     at maps.m.at.a(Unknown Source)
01-28 16:46:01.885: E/AndroidRuntime(8157):     at maps.a.bq.a(Unknown Source)
01-28 16:46:01.885: E/AndroidRuntime(8157):     at maps.a.w.a(Unknown Source)
01-28 16:46:01.885: E/AndroidRuntime(8157):     at maps.a.w.a(Unknown Source)
01-28 16:46:01.885: E/AndroidRuntime(8157):     at maps.a.w.a(Unknown Source)
01-28 16:46:01.885: E/AndroidRuntime(8157):     at maps.a.ba.m(Unknown Source)
01-28 16:46:01.885: E/AndroidRuntime(8157):     at maps.a.ba.run(Unknown Source)
01-28 16:46:01.916: W/ActivityManager(1089):   Force finishing activity
```

I thought it was something with my implementation, but in fact, I am using less heap memory now - its just this that is chugging away at it.

ni...@gmail.com <ni...@gmail.com> #5

Having exactly the same issue.
http://stackoverflow.com/questions/14523949/google-maps-android-api-v2-supportmapfragment-memory-leak#comment20254946_14523949

rc...@gmail.com <rc...@gmail.com> #6

Memory issues (and performance problems especially on low end phones) make the V2 completely unusable ... I'm having no other choices than revert my application back to V1 maps. Not will be too clever if the V2 aren't fixed and made lighter...

[Deleted User] <[Deleted User]> #7

Are you holding on to any object's beyond the lifetime of the View? The third paragraph in the MapFragment docs mention a memory leak issue.

https://developers.google.com/maps/documentation/android/reference/com/google/android/gms/maps/MapFragment

lu...@gmail.com <lu...@gmail.com> #8

[Comment deleted]

[Deleted User] <[Deleted User]> #9

We've seen the same issue on some lower end devices. We're definitely not holding any references to GoogleMap objects as suggested by comment #7:

```
01-31 14:36:38.410: E/dalvikvm-heap(15339): Out of memory on a 249752-byte allocation.
01-31 14:36:38.430: E/dalvikvm(15339): Out of memory: Heap Size=30791KB, Allocated=26955KB, Bitmap Size=953KB, Limit=32768KB
01-31 14:36:38.430: E/dalvikvm(15339): Extra info: Footprint=30727KB, Allowed Footprint=30791KB, Trimmed=2040KB
01-31 14:36:38.570: E/AndroidRuntime(15339): FATAL EXCEPTION: vts_com.banksimple
01-31 14:36:38.570: E/AndroidRuntime(15339): java.lang.OutOfMemoryError: (Heap Size=30791KB, Allocated=26955KB, Bitmap Size=953KB)
01-31 14:36:38.570: E/AndroidRuntime(15339): 	at maps.bb.d.a(Unknown Source)
01-31 14:36:38.570: E/AndroidRuntime(15339): 	at maps.s.ap.a(Unknown Source)
01-31 14:36:38.570: E/AndroidRuntime(15339): 	at maps.p.g.a(Unknown Source)
01-31 14:36:38.570: E/AndroidRuntime(15339): 	at maps.ad.j.b(Unknown Source)
01-31 14:36:38.570: E/AndroidRuntime(15339): 	at maps.ak.e.a(Unknown Source)
01-31 14:36:38.570: E/AndroidRuntime(15339): 	at maps.ak.e.b(Unknown Source)
01-31 14:36:38.570: E/AndroidRuntime(15339): 	at maps.ak.e.a(Unknown Source)
01-31 14:36:38.570: E/AndroidRuntime(15339): 	at maps.ak.o.handleMessage(Unknown Source)
01-31 14:36:38.570: E/AndroidRuntime(15339): 	at android.os.Handler.dispatchMessage(Handler.java:99)
01-31 14:36:38.570: E/AndroidRuntime(15339): 	at android.os.Looper.loop(Looper.java:150)
01-31 14:36:38.570: E/AndroidRuntime(15339): 	at maps.ak.e.j_(Unknown Source)
01-31 14:36:38.570: E/AndroidRuntime(15339): 	at maps.ak.ad.j_(Unknown Source)
01-31 14:36:38.570: E/AndroidRuntime(15339): 	at maps.bb.l.b(Unknown Source)
01-31 14:36:38.570: E/AndroidRuntime(15339): 	at maps.bb.l.run(Unknown Source)
```

kw...@gmail.com <kw...@gmail.com> #10

I am not. I clear the map and clear my reference to the marks and just pan and it continues to grow. You can open the basic map in the demo and pan around and watch the heap grow.

lu...@gmail.com <lu...@gmail.com> #11

I've seen even worse. My app terminates silently on startup, during the SupportMapFragment initialization, but only in Portrait mode. It launches ok in Landscape mode, and it survives rotatic

lu...@gmail.com <lu...@gmail.com> #12

This is what I see just before it crashes (well, exits - without crash)

```
01-31 19:27:05.307: I/dalvikvm(21111): Turning on JNI app bug workarounds for target SDK version 8...
01-31 19:27:05.323: D/ActivityThread(21111): setTargetHeapUtilization:0.25
01-31 19:27:05.323: D/ActivityThread(21111): setTargetHeapIdealFree:8388608
01-31 19:27:05.323: D/ActivityThread(21111): setTargetHeapConcurrentStart:2097152
```

cb...@google.com <cb...@google.com> #13

This issue will be fixed in the next release of the Maps Android API.

dh...@gmail.com <dh...@gmail.com> #14

Is there an approximate ETA for the next release?

cb...@google.com <cb...@google.com> #15

No ETA, sorry - I'll update this issue when the release is out. You can also subscribe to the notification group:
https://groups.google.com/group/google-maps-android-api-notify

st...@gmail.com <st...@gmail.com> #16

Would be great to see the fix released soon. Out of Memory crashes are the biggest source of trouble we're seeing since we released our app earlier this week. (We didn't see this problem wi

ja...@gmail.com <ja...@gmail.com> #17

When i adding and removing markers on the map,After some time the app will crashes.
The response in logcat is out of memory error with bitmap size

```
02-21 06:04:27.670: E/AndroidRuntime(1892): FATAL EXCEPTION: vts_com.rapidBizApps.mapamine
02-21 06:04:27.670: E/AndroidRuntime(1892): java.lang.OutOfMemoryError
02-21 06:04:27.670: E/AndroidRuntime(1892):     at maps.bb.d.a(Unknown Source)
02-21 06:04:27.670: E/AndroidRuntime(1892):     at maps.s.ap.a(Unknown Source)
02-21 06:04:27.670: E/AndroidRuntime(1892):     at maps.ak.r.a(Unknown Source)
02-21 06:04:27.670: E/AndroidRuntime(1892):     at maps.ak.e.a(Unknown Source)
02-21 06:04:27.670: E/AndroidRuntime(1892):     at maps.ak.e.a(Unknown Source)
02-21 06:04:27.670: E/AndroidRuntime(1892):     at maps.ak.o.handleMessage(Unknown Source)
02-21 06:04:27.670: E/AndroidRuntime(1892):     at android.os.Handler.dispatchMessage(Handler.java:99)
02-21 06:04:27.670: E/AndroidRuntime(1892):     at android.os.Looper.loop(Looper.java:130)
02-21 06:04:27.670: E/AndroidRuntime(1892):     at maps.ak.e.j_(Unknown Source)
02-21 06:04:27.670: E/AndroidRuntime(1892):     at maps.ak.ad.j_(Unknown Source)
02-21 06:04:27.670: E/AndroidRuntime(1892):     at maps.bb.l.b(Unknown Source)
02-21 06:04:27.670: E/AndroidRuntime(1892):     at maps.bb.l.run(Unknown Source)
02-21 06:04:33.260: D/dalvikvm(1892): GC_EXPLICIT freed 2148K, 53% free 5309K/11207K, external 21007K/21768K, paused 71ms
02-21 06:09:27.740: I/Process(1892): Sending signal. PID: 1892 SIG: 9
```

st...@gmail.com <st...@gmail.com> #18

Per #4, #9 and #17 above, of the 16 or so different out of memory crash report types I'm seeing (all but one of which originate in Google Maps), these two are the most common:

```
Stack Trace
________________________________
0      java.lang.OutOfMemoryError
1          at android.graphics.Bitmap.nativeCreate(Native Method)
2          at android.graphics.Bitmap.createBitmap(Bitmap.java:640)
3          at android.graphics.Bitmap.createBitmap(Bitmap.java:620)
4          at maps.r.h.a(Unknown Source)
5          at maps.cp.a.a(Unknown Source)
6          at maps.cp.a.a(Unknown Source)
7          at maps.y.y.b(Unknown Source)
8          at maps.y.y.a(Unknown Source)
9          at maps.y.bg.a(Unknown Source)
10          at maps.a.w.a(Unknown Source)
11          at maps.a.w.a(Unknown Source)
12          at maps.a.w.a(Unknown Source)
13          at maps.a.ba.m(Unknown Source)
14          at maps.a.ba.run(Unknown Source)
```

```
Stack Trace
________________________________
0      java.lang.OutOfMemoryError
1          at maps.bb.d.a(Unknown Source)
2          at maps.s.ap.a(Unknown Source)
3          at maps.ak.r.a(Unknown Source)
4          at maps.ak.e.a(Unknown Source)
5          at maps.ak.e.a(Unknown Source)
6          at maps.ak.o.handleMessage(Unknown Source)
7          at android.os.Handler.dispatchMessage(Handler.java:99)
8          at android.os.Looper.loop(Looper.java:137)
9          at maps.ak.e.j_(Unknown Source)
10          at maps.ak.ad.j_(Unknown Source)
11          at maps.bb.l.b(Unknown Source)
12          at maps.bb.l.run(Unknown Source)
```

This memory issue is the only significant production I have in the app. An expedited release of the fix would be very much appreciated.

[Deleted User] <[Deleted User]> #19

The release notes for the Feb 2013 update (https://developers.google.com/maps/documentation/android/releases#february_2013) indicate that this bug has been fixed, but it hasn't. The or

1. Get latest Maps API
2. Make a new 'maps' Android demo project
3. Rotate phone to cause activity to be destroyed and recreated and watch it leak with every rotation

Is there some other special voodoo to use the fix?

ad...@google.com <ad...@google.com> #20

What device are you running on? Can you tell us what version of Google Play services you have running on your device?

[Deleted User] <[Deleted User]> #21

Re: #20

This is on a Nexus S 4G running 2.3.7. Google Play Services was just updated to version 3.0.25 (583950-10).

cb...@google.com <cb...@google.com>

Marked as fixed, reassigned to cb...@google.com.

[Deleted User] <[Deleted User]> #22

Re: #20

I can also reproduce this on a Galaxy Nexus running 4.2.2 (same version of Play Services). With the 'Basic Map' from the 'maps' demo project it crashes with an OutOfMemoryError after noth
dump with MAT shows 97 instances of BasicMapActivity at the time of the crash.

ad...@google.com <ad...@google.com> #23

Thanks for the details. A few more questions:
- Does the memory keep rising by a similar amount every time you rotate the device, with the app ultimately crashing?
- Are you able to provide us with a hprof file so that we can investigate what is causing the memory leak on your device.

[Deleted User] <[Deleted User]> #24

Re: #25

The memory increases by about a meg on every rotation (split between internal and external memory on older devices), and the app does ultimately crash.

Attached is the dump from the Nexus S. I'll see if I can compress the dump from the Galaxy Nexus enough to attach it in another comment.

4766-OoM-NexusS4G-2.3.7.tgz
4.1 MB Download

[Deleted User] <[Deleted User]> #25

Re: #25

Here's the (already converted) hprof from the Galaxy Nexus (4.2.2) when it crashed.

While I'm happy to provide these dumps, I'm a bit worried. Are you not able to reproduce this on devices there?

4766-OoM-GalaxyNexus-4.2.2.xz
5.4 MB Download

st...@gmail.com <st...@gmail.com> #26

Tested the latest Google Play services sdk on HTC Evo 3D.

Attaching the heap dump.

android3713074625331561724.zip
8.9 MB Download

cb...@google.com <cb...@google.com> #27

Status: New (reopened)

Re-opening.

az...@gmail.com <az...@gmail.com> #28

It's nice to know that the guys at google are human too :)

an...@gmail.com <an...@gmail.com> #29

Hi,

I have tried the newer release (2013/2) on my Samsung S3 (android 4.1.2), my scenario is from Activity "A" to Activity B(Google Map v2 with satellite type) and finish Activity B , as I observed

1. The big difference with the previous release on 2012/12 is that after the activity B is finished, the "heap size" (from eclipse DDMS) is still the same but the "allocated' decreased largely. See

2. But still one thing bothers me is if I move the map largely, the heap growth very fast. Finally, it crashed when the heap size exceed the limit, 64MB on my device. Is this a problem about goo

Thanks.

st...@gmail.com <st...@gmail.com> #30

So while it looks like not quite all out of memory issues are resolved, from my crash report stats, it looks like around a 60-70% reduction in occurrences for my app, which is a decent improve

an...@gmail.com <an...@gmail.com> #31

HI,

Further explain about #31,

======================
But still one thing bothers me is if I move the map again and again and again....., the heap will grow very fast. Finally, it crashed when the heap size exceed the limit, 64MB on my device. Is th
=====================

And here is the error from Eclipse log

```
====================
03-02 00:06:06.000: E/dalvikvm-heap(14113): Out of memory on a 4194320-byte allocation.
03-02 00:06:06.015: A/libc(14113): Fatal signal 11 (SIGSEGV) at 0x0004f75c (code=1), thread 14113
====================
```

And I am really wondering why the google "Maps" app does not encounter this problem.

ce...@gmail.com <ce...@gmail.com> #32

I think this problem still happens, here is the errors from BugSense:

```
0java.lang.OutOfMemoryError
1at org.apache.harmony.luni.platform.OSMemory.malloc(Native Method)
2at org.apache.harmony.luni.platform.PlatformAddressFactory.alloc(PlatformAddressFactory.java:150)
3at java.nio.DirectByteBuffer.<init>(DirectByteBuffer.java:66)
4at java.nio.ReadWriteDirectByteBuffer.<init>(ReadWriteDirectByteBuffer.java:51)
5at java.nio.BufferFactory.newDirectByteBuffer(BufferFactory.java:93)
6at java.nio.ByteBuffer.allocateDirect(ByteBuffer.java:68)
7at maps.al.m.a(Unknown Source)
8at maps.al.m.c(Unknown Source)
9at maps.al.e.c(Unknown Source)
10at maps.al.e.d(Unknown Source)
11at maps.l.an.b(Unknown Source)
12at maps.y.ab.a(Unknown Source)
13at maps.af.v.a(Unknown Source)
14at maps.af.v.a(Unknown Source)
15at maps.af.v.a(Unknown Source)
16at maps.p.q.m(Unknown Source)
17at maps.p.q.run(Unknown Source)
```

```
App Version     Country Phone   OS Version      Wifi On Mobile Net On   Show All
1.2     ES      HTC Desire      2.2.2
1.2     ES      HTC Desire      2.2.2
```

ju...@gmail.com <ju...@gmail.com> #33

yes, it stills happens.

I´m trying in one Samsung S3 and is getting out of memory...

jm...@gmail.com <jm...@gmail.com> #34

This is really horrible on my Verizon Samsung Galaxy S3. I tried turning on the large heap in hopes that my app would go longer without crashing, but all that happened was the app appeared
gc'ing, before finally crashing. The actual useable time of my app did not increase at all.

```
03-09 03:06:20.486: D/dalvikvm(1861): GC_CONCURRENT freed 5755K, 31% free 14480K/20931K, paused 13ms+39ms, total 152ms
03-09 03:06:23.209: D/dalvikvm(1861): GC_CONCURRENT freed 4813K, 29% free 15007K/20931K, paused 18ms+4ms, total 125ms
03-09 03:06:23.990: D/dalvikvm(1861): GC_FOR_ALLOC freed 2472K, 25% free 15885K/20931K, paused 46ms, total 48ms
03-09 03:06:24.150: D/dalvikvm(1861): GC_CONCURRENT freed 9K, 12% free 28052K/31623K, paused 3ms+61ms, total 138ms
03-09 03:06:24.751: D/dalvikvm(1861): GC_CONCURRENT freed 14K, 3% free 36170K/36935K, paused 13ms+78ms, total 262ms
03-09 03:06:24.751: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 114ms
03-09 03:06:24.751: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 118ms
03-09 03:06:25.391: D/dalvikvm(1861): GC_CONCURRENT freed 12K, 2% free 44234K/44999K, paused 3ms+79ms, total 322ms
03-09 03:06:25.391: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 230ms
03-09 03:06:26.182: D/dalvikvm(1861): GC_CONCURRENT freed 326K, 3% free 51976K/53063K, paused 5ms+97ms, total 452ms
```

```
03-09 03:06:26.182: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 360ms
03-09 03:06:27.033: D/dalvikvm(1861): GC_CONCURRENT freed <1K, 2% free 60046K/60807K, paused 2ms+137ms, total 515ms
03-09 03:06:27.033: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 424ms
03-09 03:06:27.884: D/dalvikvm(1861): GC_CONCURRENT freed 0K, 2% free 68110K/68871K, paused 3ms+112ms, total 542ms
03-09 03:06:27.884: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 451ms
03-09 03:06:28.855: D/dalvikvm(1861): GC_CONCURRENT freed 0K, 1% free 76174K/76935K, paused 5ms+121ms, total 615ms
03-09 03:06:28.855: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 522ms
03-09 03:06:29.866: D/dalvikvm(1861): GC_CONCURRENT freed 0K, 1% free 84238K/84999K, paused 3ms+131ms, total 687ms
03-09 03:06:29.866: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 595ms
03-09 03:06:30.947: D/dalvikvm(1861): GC_CONCURRENT freed 0K, 1% free 92302K/93063K, paused 3ms+150ms, total 765ms
03-09 03:06:30.947: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 674ms
03-09 03:06:32.118: D/dalvikvm(1861): GC_CONCURRENT freed <1K, 1% free 100366K/101127K, paused 2ms+160ms, total 843ms
03-09 03:06:32.118: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 753ms
03-09 03:06:33.400: D/dalvikvm(1861): GC_CONCURRENT freed 0K, 1% free 108430K/109191K, paused 4ms+172ms, total 935ms
03-09 03:06:33.400: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 843ms
03-09 03:06:34.721: D/dalvikvm(1861): GC_CONCURRENT freed 0K, 1% free 116494K/117255K, paused 3ms+182ms, total 994ms
03-09 03:06:34.721: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 903ms
03-09 03:06:36.153: D/dalvikvm(1861): GC_CONCURRENT freed 0K, 1% free 124558K/125319K, paused 5ms+196ms, total 1090ms
03-09 03:06:36.153: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 997ms
03-09 03:06:37.634: D/dalvikvm(1861): GC_CONCURRENT freed 0K, 1% free 132622K/133383K, paused 3ms+208ms, total 1149ms
03-09 03:06:37.634: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1058ms
03-09 03:06:39.206: D/dalvikvm(1861): GC_CONCURRENT freed 0K, 1% free 140686K/141447K, paused 3ms+218ms, total 1237ms
03-09 03:06:39.206: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1146ms
03-09 03:06:40.838: D/dalvikvm(1861): GC_CONCURRENT freed 0K, 1% free 148750K/149511K, paused 2ms+231ms, total 1308ms
03-09 03:06:40.838: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1218ms
03-09 03:06:42.419: D/dalvikvm(1861): GC_CONCURRENT freed 2432K, 2% free 154482K/157575K, paused 6ms+16ms, total 1180ms
03-09 03:06:42.419: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 968ms
03-09 03:06:44.281: D/dalvikvm(1861): GC_CONCURRENT freed 6893K, 5% free 155301K/162887K, paused 6ms+32ms, total 1218ms
03-09 03:06:44.281: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 966ms
03-09 03:06:46.043: D/dalvikvm(1861): GC_FOR_ALLOC freed 4663K, 5% free 155882K/162887K, paused 1192ms, total 1192ms
03-09 03:06:47.895: D/dalvikvm(1861): GC_CONCURRENT freed 6990K, 5% free 156755K/164423K, paused 5ms+60ms, total 1312ms
03-09 03:06:47.895: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1040ms
03-09 03:06:49.107: D/dalvikvm(1861): GC_FOR_ALLOC freed <1K, 5% free 156755K/164423K, paused 1201ms, total 1201ms
03-09 03:06:50.718: D/dalvikvm(1861): GC_CONCURRENT freed 6914K, 5% free 158064K/165639K, paused 6ms+76ms, total 1286ms
03-09 03:06:50.718: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 972ms
03-09 03:06:52.500: D/dalvikvm(1861): GC_FOR_ALLOC freed 3575K, 5% free 158064K/165639K, paused 1202ms, total 1202ms
03-09 03:06:54.272: D/dalvikvm(1861): GC_CONCURRENT freed 11498K, 8% free 160056K/172231K, paused 13ms+343ms, total 1635ms
03-09 03:06:56.194: D/dalvikvm(1861): GC_FOR_ALLOC freed 4227K, 8% free 160028K/172231K, paused 1270ms, total 1270ms
03-09 03:06:57.956: D/dalvikvm(1861): GC_CONCURRENT freed 14082K, 11% free 162973K/181127K, paused 18ms+391ms, total 1732ms
03-09 03:06:57.956: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 495ms
03-09 03:07:00.469: D/dalvikvm(1861): GC_CONCURRENT freed 8191K, 11% free 162973K/181127K, paused 3ms+177ms, total 1585ms
03-09 03:07:00.469: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1282ms
03-09 03:07:01.970: D/dalvikvm(1861): GC_FOR_ALLOC freed 1314K, 11% free 162973K/181127K, paused 1306ms, total 1306ms
03-09 03:07:03.872: D/dalvikvm(1861): GC_CONCURRENT freed 17027K, 14% free 167391K/194439K, paused 17ms+451ms, total 1846ms
03-09 03:07:03.872: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 617ms
03-09 03:07:05.604: D/dalvikvm(1861): GC_FOR_ALLOC freed 13433K, 20% free 156730K/194439K, paused 1261ms, total 1261ms
03-09 03:07:07.276: D/dalvikvm(1861): GC_CONCURRENT freed 1K, 12% free 171487K/194439K, paused 14ms+352ms, total 1669ms
03-09 03:07:16.266: D/dalvikvm(1861): GC_CONCURRENT freed 20822K, 19% free 158857K/194439K, paused 6ms+35ms, total 1367ms
03-09 03:07:16.266: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 882ms
03-09 03:07:19.319: D/dalvikvm(1861): GC_CONCURRENT freed 2352K, 16% free 164529K/194439K, paused 4ms+41ms, total 1481ms
03-09 03:07:19.319: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1035ms
03-09 03:07:22.422: D/dalvikvm(1861): GC_CONCURRENT freed 2117K, 13% free 170604K/194439K, paused 13ms+248ms, total 1746ms
03-09 03:07:22.422: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1211ms
03-09 03:07:25.265: D/dalvikvm(1861): GC_CONCURRENT freed 2184K, 10% free 176611K/194439K, paused 6ms+44ms, total 1632ms
03-09 03:07:25.265: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1137ms
03-09 03:07:28.479: D/dalvikvm(1861): GC_CONCURRENT freed 1921K, 6% free 182882K/194439K, paused 14ms+39ms, total 1732ms
03-09 03:07:28.479: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1148ms
03-09 03:07:31.622: D/dalvikvm(1861): GC_CONCURRENT freed 1522K, 4% free 187918K/194439K, paused 13ms+23ms, total 1711ms
03-09 03:07:31.622: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1542ms
03-09 03:07:34.465: D/dalvikvm(1861): GC_CONCURRENT freed 2458K, 2% free 193654K/196743K, paused 3ms+52ms, total 1812ms
03-09 03:07:34.465: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1287ms
03-09 03:07:37.879: D/dalvikvm(1861): GC_CONCURRENT freed 1934K, 2% free 199939K/202503K, paused 6ms+49ms, total 1931ms
03-09 03:07:37.879: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1373ms
03-09 03:07:41.282: D/dalvikvm(1861): GC_CONCURRENT freed 1917K, 2% free 206227K/208775K, paused 13ms+52ms, total 1963ms
03-09 03:07:41.282: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1314ms
03-09 03:07:44.976: D/dalvikvm(1861): GC_CONCURRENT freed 1929K, 2% free 212551K/215111K, paused 5ms+53ms, total 2162ms
03-09 03:07:44.976: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1620ms
03-09 03:07:48.610: D/dalvikvm(1861): GC_CONCURRENT freed 1917K, 2% free 218836K/221383K, paused 13ms+58ms, total 2173ms
03-09 03:07:48.610: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1588ms
03-09 03:07:51.553: D/dalvikvm(1861): GC_CONCURRENT freed 2838K, 2% free 223520K/227719K, paused 4ms+78ms, total 2281ms
03-09 03:07:51.553: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 359ms
03-09 03:07:51.553: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1897ms
03-09 03:07:55.377: D/dalvikvm(1861): GC_CONCURRENT freed 1952K, 2% free 229809K/232391K, paused 6ms+67ms, total 2285ms
03-09 03:07:55.377: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1751ms
03-09 03:07:59.311: D/dalvikvm(1861): GC_CONCURRENT freed 1922K, 2% free 236111K/238663K, paused 4ms+69ms, total 2435ms
03-09 03:07:59.311: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1886ms
03-09 03:08:03.356: D/dalvikvm(1861): GC_CONCURRENT freed 1915K, 2% free 242390K/244935K, paused 13ms+71ms, total 2494ms
03-09 03:08:03.356: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1883ms
03-09 03:08:04.377: D/AbsListView(1861): Get MotionRecognitionManager
03-09 03:08:07.500: D/dalvikvm(1861): GC_CONCURRENT freed 1927K, 2% free 248713K/251271K, paused 13ms+72ms, total 2604ms
03-09 03:08:07.500: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 1999ms
03-09 03:08:11.665: D/dalvikvm(1861): GC_CONCURRENT freed 1917K, 1% free 254996K/257543K, paused 4ms+74ms, total 2668ms
03-09 03:08:11.665: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 2135ms
03-09 03:08:15.569: D/dalvikvm(1861): GC_CONCURRENT freed 1499K, 1% free 259958K/262087K, paused 4ms+99ms, total 2803ms
03-09 03:08:15.569: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 2276ms
03-09 03:08:18.282: D/dalvikvm(1861): GC_FOR_ALLOC freed <1K, 1% free 259958K/262087K, paused 2708ms, total 2708ms
```

```
03-09 03:08:21.065: D/dalvikvm(1861): GC_BEFORE_OOM freed 72K, 1% free 259885K/262087K, paused 2781ms, total 2781ms
03-09 03:08:23.758: D/dalvikvm(1861): GC_CONCURRENT freed 25K, 1% free 259968K/262087K, paused 14ms+55ms, total 2707ms
03-09 03:08:23.758: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 2648ms
03-09 03:08:26.440: D/dalvikvm(1861): GC_FOR_ALLOC freed <1K, 1% free 259968K/262087K, paused 2672ms, total 2672ms
03-09 03:08:29.173: D/dalvikvm(1861): GC_BEFORE_OOM freed 0K, 1% free 259968K/262087K, paused 2735ms, total 2735ms
03-09 03:08:29.173: E/dalvikvm-heap(1861): Out of memory on a 48-byte allocation.
03-09 03:08:29.173: I/dalvikvm(1861): "GLThread 4219" prio=5 tid=19 RUNNABLE
03-09 03:08:29.173: I/dalvikvm(1861):   | group="main" sCount=0 dsCount=0 obj=0x42029118 self=0x56d32c30
03-09 03:08:29.173: I/dalvikvm(1861):   | sysTid=1892 nice=1 sched=0/0 cgrp=apps handle=1456681088
03-09 03:08:29.173: I/dalvikvm(1861):   | schedstat=( 0 0 0 ) utm=7693 stm=163 core=1
03-09 03:08:29.173: I/dalvikvm(1861):   at maps.f.cs.a((null):~-1)
03-09 03:08:29.173: I/dalvikvm(1861):   at maps.t.al.<init>((null):-1)
03-09 03:08:29.173: I/dalvikvm(1861):   at maps.t.ah.<init>((null):-1)
03-09 03:08:29.173: I/dalvikvm(1861):   at maps.t.ah.b((null):-1)
03-09 03:08:29.173: I/dalvikvm(1861):   at maps.bk.b.a((null):-1)
03-09 03:08:29.173: I/dalvikvm(1861):   at maps.y.ab.a((null):-1)
03-09 03:08:29.173: I/dalvikvm(1861):   at maps.y.ab.a((null):-1)
03-09 03:08:29.173: I/dalvikvm(1861):   at maps.af.v.a((null):-1)
03-09 03:08:29.173: I/dalvikvm(1861):   at maps.af.v.a((null):-1)
03-09 03:08:29.173: I/dalvikvm(1861):   at maps.af.v.a((null):-1)
03-09 03:08:29.173: I/dalvikvm(1861):   at maps.p.q.m((null):-1)
03-09 03:08:29.173: I/dalvikvm(1861):   at maps.p.q.run((null):-1)
03-09 03:08:32.387: D/dalvikvm(1861): GC_CONCURRENT freed 0K, 1% free 259968K/262087K, paused 14ms+13ms, total 2692ms
03-09 03:08:32.387: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 2678ms
03-09 03:08:35.130: D/dalvikvm(1861): GC_FOR_ALLOC freed 0K, 1% free 259968K/262087K, paused 2745ms, total 2745ms
03-09 03:08:38.003: D/dalvikvm(1861): GC_BEFORE_OOM freed 0K, 1% free 259968K/262087K, paused 2873ms, total 2873ms
03-09 03:08:38.003: E/dalvikvm-heap(1861): Out of memory on a 112-byte allocation.
03-09 03:08:38.003: I/dalvikvm(1861): "GLThread 4219" prio=5 tid=19 RUNNABLE
03-09 03:08:38.003: I/dalvikvm(1861):   | group="main" sCount=0 dsCount=0 obj=0x42029118 self=0x56d32c30
03-09 03:08:38.003: I/dalvikvm(1861):   | sysTid=1892 nice=1 sched=0/0 cgrp=apps handle=1456681088
03-09 03:08:38.003: I/dalvikvm(1861):   | schedstat=( 0 0 0 ) utm=8230 stm=178 core=0
03-09 03:08:38.003: I/dalvikvm(1861):   at java.lang.Throwable.nativeFillInStackTrace(Native Method)
03-09 03:08:38.003: I/dalvikvm(1861):   at java.lang.Throwable.fillInStackTrace(Throwable.java:160)
03-09 03:08:38.003: I/dalvikvm(1861):   at java.lang.Throwable.<init>(Throwable.java:83)
03-09 03:08:38.003: I/dalvikvm(1861):   at java.lang.Error.<init>(Error.java:37)
03-09 03:08:38.003: I/dalvikvm(1861):   at java.lang.VirtualMachineError.<init>(VirtualMachineError.java:35)
03-09 03:08:38.003: I/dalvikvm(1861):   at java.lang.OutOfMemoryError.<init>(OutOfMemoryError.java:33)
03-09 03:08:38.003: I/dalvikvm(1861):   at maps.f.cs.a((null):-1)
03-09 03:08:38.003: I/dalvikvm(1861):   at maps.t.al.<init>((null):-1)
03-09 03:08:38.003: I/dalvikvm(1861):   at maps.t.ah.<init>((null):-1)
03-09 03:08:38.003: I/dalvikvm(1861):   at maps.t.ah.b((null):-1)
03-09 03:08:38.003: I/dalvikvm(1861):   at maps.bk.b.a((null):-1)
03-09 03:08:38.003: I/dalvikvm(1861):   at maps.y.ab.a((null):-1)
03-09 03:08:38.003: I/dalvikvm(1861):   at maps.y.ab.a((null):-1)
03-09 03:08:38.003: I/dalvikvm(1861):   at maps.af.v.a((null):-1)
03-09 03:08:38.003: I/dalvikvm(1861):   at maps.af.v.a((null):-1)
03-09 03:08:38.003: I/dalvikvm(1861):   at maps.af.v.a((null):-1)
03-09 03:08:38.003: I/dalvikvm(1861):   at maps.p.q.m((null):-1)
03-09 03:08:38.003: I/dalvikvm(1861):   at maps.p.q.run((null):-1)
03-09 03:08:38.003: W/dalvikvm(1861): Exception thrown (Ljava/lang/OutOfMemoryError;) while throwing internal exception (Ljava/lang/OutOfMemoryError;)
03-09 03:08:41.056: D/dalvikvm(1861): GC_CONCURRENT freed 24289K, 11% free 235679K/262087K, paused 14ms+12ms, total 2529ms
03-09 03:08:41.056: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 2515ms
03-09 03:08:41.056: W/dalvikvm(1861): threadid=19: thread exiting with uncaught exception (group=0x41274438)
03-09 03:08:41.056: D/dalvikvm(1861): WAIT_FOR_CONCURRENT_GC blocked 2516ms
03-09 03:08:41.056: E/AndroidRuntime(1861): FATAL EXCEPTION: GLThread 4219
03-09 03:08:41.056: E/AndroidRuntime(1861): java.lang.OutOfMemoryError: [memory exhausted]
03-09 03:08:41.056: E/AndroidRuntime(1861):     at dalvik.system.NativeStart.main(Native Method)
```

Without the large heap the output is much the same but with less gc thrashing.

Are more dumps required, and if so, how should I capture one?

[Deleted User] <[Deleted User]> #35

I'm also experiencing a problem that looks to be similar from this stack trace:

```
java.lang.OutOfMemoryError
      at android.graphics.Bitmap.nativeCreate(Native Method)
      at android.graphics.Bitmap.createBitmap(Bitmap.java:640)
      at android.graphics.Bitmap.createBitmap(Bitmap.java:620)
      at maps.s.h.a(Unknown Source)
      at maps.cr.a.a(Unknown Source)
      at maps.cr.a.a(Unknown Source)
      at maps.cr.a.b(Unknown Source)
      at maps.l.m.a(Unknown Source)
      at maps.l.at.a(Unknown Source)
      at maps.y.ab.a(Unknown Source)
      at maps.af.v.a(Unknown Source)
      at maps.af.v.a(Unknown Source)
      at maps.af.v.a(Unknown Source)
      at maps.p.q.m(Unknown Source)
      at maps.p.q.run(Unknown Source)
```

ad...@google.com <ad...@google.com> #36

I have been looking further into this and thank you very much for the HPROF dumps.

@studentay1 - it appears that your heap dump illustrates the following more specific bug https://code.google.com/p/gmaps-api-issues/issues/detail?id=4703. There is a workaround mentio

@abuchanan - We have tried hard to reproduce the memory leak in the hprof dumps that you have provided but have not been able to. Continually rotating the device while showing the Basic
We have tried on a Nexus S running 2.3 and a Galaxy Nexus running 4.2 and have not had any memory issues. Are the devices you are using running stock system images or a special ROM?

If anyone is still experiencing memory issues, of most use to us would be an hprof dump as well as the source code of a minimal example that produced the memory leak. If you are unable t
(in addition to the hprof dump) that would also be useful. Unfortunately stack traces don't really help us identify the cause of the issue because where the crash occurs doesn't indicate what

[Deleted User] <[Deleted User]> #37

@admo - I realized belatedly that I'd captured those dumps with the debugger attached (oops!). I haven't see any leaks running regularly. I would've chimed in earlier, but before I realized my
and everyone else had chimed in with similar problems. I assumed there must have been some lingering problem that I was just fortunate enough to avoid. I hope this didn't cause y'all too n

ad...@google.com <ad...@google.com> #38

@abuchanan - that's ok. Thanks for confirming.

It appears that https://code.google.com/p/gmaps-api-issues/issues/detail?id=4703 is the only confirmed memory issue. If other memory issues are still reproducible, please provide us with

jm...@gmail.com <jm...@gmail.com> #39

When I last posted here I was seeing this with even the most trivial of map apps. Now I find that it is no longer crashing, and Google Play Services is at the same version it was before? Assu
the version number, the only possible explanation is that the possibility for this behavior still exists, but it is somehow tied to the responses the app is receiving from Google's servers.

jm...@gmail.com <jm...@gmail.com> #40

My problem was certainly not connected to the marker feature.

[Deleted User] <[Deleted User]> #41

I'm not seeing memory "leaks" exactly, but the maps code is grabbing huge chunks of memory:

```
Android: 2.3.5
Manufacturer: HTC
Model: PC36100
Date: Mon Apr 01 11:17:25 PDT 2013

java.lang.OutOfMemoryError: bitmap size exceeds VM budget(Heap Size=13255KB, Allocated=9146KB, Bitmap Size=19553KB)
      at android.graphics.Bitmap.nativeCreate(Native Method)
      at android.graphics.Bitmap.createBitmap(Bitmap.java:695)
      at maps.s.h.a(Unknown Source)
      at maps.cr.a.a(Unknown Source)
      at maps.cr.a.a(Unknown Source)
      at maps.cr.a.b(Unknown Source)
      at maps.l.m.a(Unknown Source)
      at maps.l.at.a(Unknown Source)
      at maps.y.ab.a(Unknown Source)
      at maps.af.v.a(Unknown Source)
      at maps.af.v.a(Unknown Source)
      at maps.af.v.a(Unknown Source)
      at maps.p.q.m(Unknown Source)
      at maps.p.q.run(Unknown Source)
```

This looks like the same trace others are reporting but it includes the size of the bitmap (20MB! Can that be right?). The model number suggests this is an Evo 4G, which has a 480x800 scre
only, so I'm not sure why the crash suggests a 20MB bitmap is being generated. The maps in this app are 280dp tall and the width of the device. No more than two markers are displayed tho

I'll see if I can get ahold of one of these devices to generate an hprof. The app isn't crashing on any generation of Nexus device (regularly testing on Nexus One, S, Galaxy Nexus, and Nexus 4
on, but I'll also look more closely at memory usage to see if there are any obvious patterns.

te...@gmail.com <te...@gmail.com> #42

Same issue facing huge amout of memory leak and that's why get outofmemory error and app got crash.

I have tested
SAMSUNG GALAXY X-Cover model GT-S5690 Android vers. 2.3.6
Sony Xperia.
And emulator with 4.2 os

Please suggest a solution for it I am stucking since last few days and than able to find the crash is beacause of the SupportMapFragment memory leak

Thanks.

te...@gmail.com <te...@gmail.com> #43

Same issue facing huge amout of memory leak and that's why get outofmemory error and app got crash.

I have tested
SAMSUNG GALAXY X-Cover model GT-S5690 Android vers. 2.3.6
Sony Xperia.
And emulator with 4.2 os

Please suggest a solution for it I am stucking since last few days and than able to find the crash is beacause of the SupportMapFragment memory leak

```
04-09 13:04:06.589: E/dalvikvm-heap(1612): Out of memory on a 1420816-byte allocation.
04-09 13:04:06.589: I/dalvikvm(1612): "main" prio=5 tid=1 RUNNABLE
04-09 13:04:06.589: I/dalvikvm(1612):   | group="main" sCount=0 dsCount=0 obj=0x40a14568 self=0x2a00b9e0
04-09 13:04:06.589: I/dalvikvm(1612):   | sysTid=1612 nice=0 sched=0/0 cgrp=apps handle=1073870640
04-09 13:04:06.589: I/dalvikvm(1612):   | schedstat=( 8049868940 10666856779 4014 ) utm=709 stm=95 core=0
04-09 13:04:06.589: I/dalvikvm(1612):   at android.graphics.Bitmap.nativeCreate(Native Method)
04-09 13:04:06.589: I/dalvikvm(1612):   at android.graphics.Bitmap.createBitmap(Bitmap.java:640)
04-09 13:04:06.589: I/dalvikvm(1612):   at android.graphics.Bitmap.createBitmap(Bitmap.java:620)
04-09 13:04:06.589: I/dalvikvm(1612):   at android.view.View.buildDrawingCache(View.java:12653)
04-09 13:04:06.600: I/dalvikvm(1612):   at android.view.ViewGroup.onAnimationStart(ViewGroup.java:2637)
04-09 13:04:06.600: I/dalvikvm(1612):   at android.view.View.drawAnimation(View.java:12954)
04-09 13:04:06.600: I/dalvikvm(1612):   at android.view.View.draw(View.java:13096)
04-09 13:04:06.600: I/dalvikvm(1612):   at android.view.ViewGroup.drawChild(ViewGroup.java:2929)
04-09 13:04:06.600: I/dalvikvm(1612):   at android.view.ViewGroup.dispatchDraw(ViewGroup.java:2799)
04-09 13:04:06.600: I/dalvikvm(1612):   at android.view.View.draw(View.java:13340)
04-09 13:04:06.600: I/dalvikvm(1612):   at android.view.ViewGroup.drawChild(ViewGroup.java:2929)
04-09 13:04:06.600: I/dalvikvm(1612):   at android.view.ViewGroup.dispatchDraw(ViewGroup.java:2799)
04-09 13:04:06.600: I/dalvikvm(1612):   at android.view.View.draw(View.java:13340)
04-09 13:04:06.600: I/dalvikvm(1612):   at android.view.ViewGroup.drawChild(ViewGroup.java:2929)
04-09 13:04:06.600: I/dalvikvm(1612):   at android.view.ViewGroup.dispatchDraw(ViewGroup.java:2799)
04-09 13:04:06.600: I/dalvikvm(1612):   at android.view.View.draw(View.java:13461)
04-09 13:04:06.609: I/dalvikvm(1612):   at android.widget.FrameLayout.draw(FrameLayout.java:467)
04-09 13:04:06.609: I/dalvikvm(1612):   at android.view.View.draw(View.java:13342)
04-09 13:04:06.609: I/dalvikvm(1612):   at android.view.ViewGroup.drawChild(ViewGroup.java:2929)
04-09 13:04:06.609: I/dalvikvm(1612):   at android.view.ViewGroup.dispatchDraw(ViewGroup.java:2799)
04-09 13:04:06.609: I/dalvikvm(1612):   at android.view.View.draw(View.java:13340)
04-09 13:04:06.609: I/dalvikvm(1612):   at android.view.ViewGroup.drawChild(ViewGroup.java:2929)
04-09 13:04:06.609: I/dalvikvm(1612):   at android.view.ViewGroup.dispatchDraw(ViewGroup.java:2799)
04-09 13:04:06.609: I/dalvikvm(1612):   at android.view.View.draw(View.java:13340)
04-09 13:04:06.609: I/dalvikvm(1612):   at android.view.ViewGroup.drawChild(ViewGroup.java:2929)
04-09 13:04:06.609: I/dalvikvm(1612):   at android.view.ViewGroup.dispatchDraw(ViewGroup.java:2799)
04-09 13:04:06.609: I/dalvikvm(1612):   at android.view.View.draw(View.java:13461)
04-09 13:04:06.619: I/dalvikvm(1612):   at android.widget.FrameLayout.draw(FrameLayout.java:467)
04-09 13:04:06.619: I/dalvikvm(1612):   at com.android.internal.policy.impl.PhoneWindow$DecorView.draw(PhoneWindow.java:2183)
04-09 13:04:06.619: I/dalvikvm(1612):   at android.view.ViewRootImpl.drawSoftware(ViewRootImpl.java:2258)
04-09 13:04:06.619: I/dalvikvm(1612):   at android.view.ViewRootImpl.draw(ViewRootImpl.java:2153)
04-09 13:04:06.619: I/dalvikvm(1612):   at android.view.ViewRootImpl.performDraw(ViewRootImpl.java:2021)
04-09 13:04:06.619: I/dalvikvm(1612):   at android.view.ViewRootImpl.performTraversals(ViewRootImpl.java:1832)
04-09 13:04:06.619: I/dalvikvm(1612):   at android.view.ViewRootImpl.doTraversal(ViewRootImpl.java:1000)
04-09 13:04:06.619: I/dalvikvm(1612):   at android.view.ViewRootImpl$TraversalRunnable.run(ViewRootImpl.java:4214)
04-09 13:04:06.619: I/dalvikvm(1612):   at android.view.Choreographer$CallbackRecord.run(Choreographer.java:725)
04-09 13:04:06.619: I/dalvikvm(1612):   at android.view.Choreographer.doCallbacks(Choreographer.java:555)
04-09 13:04:06.619: I/dalvikvm(1612):   at android.view.Choreographer.doFrame(Choreographer.java:525)
04-09 13:04:06.629: I/dalvikvm(1612):   at android.view.Choreographer$FrameDisplayEventReceiver.run(Choreographer.java:711)
04-09 13:04:06.629: I/dalvikvm(1612):   at android.os.Handler.handleCallback(Handler.java:615)
04-09 13:04:06.629: I/dalvikvm(1612):   at android.os.Handler.dispatchMessage(Handler.java:92)
04-09 13:04:06.629: I/dalvikvm(1612):   at android.os.Looper.loop(Looper.java:137)
04-09 13:04:06.629: I/dalvikvm(1612):   at android.app.ActivityThread.main(ActivityThread.java:4745)
04-09 13:04:06.629: I/dalvikvm(1612):   at java.lang.reflect.Method.invokeNative(Native Method)
04-09 13:04:06.629: I/dalvikvm(1612):   at java.lang.reflect.Method.invoke(Method.java:511)
04-09 13:04:06.629: I/dalvikvm(1612):   at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:786)
04-09 13:04:06.629: I/dalvikvm(1612):   at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:553)
04-09 13:04:06.639: I/dalvikvm(1612):   at dalvik.system.NativeStart.main(Native Method)
04-09 13:04:06.969: I/dalvikvm-heap(1612): Clamp target GC heap from 16.275MB to 16.000MB
```

Thanks.

ma...@gmail.com <ma...@gmail.com> #44

Comment has been deleted.

Message last modified on May 7, 2021 10:28PM

ya...@drosidis.gr <ya...@drosidis.gr> #45

I believe I have isolated a memory leak issue of the Map API.

The heap usage increases by about 85KB every time the activity becomes visible again, after one of the following has happened:
- Phone screen turns off (eg after pressing the power button).
- The user exits the app pressing the Home button.

The leak does NOT occur on screen rotate, or when exiting by "back" button.

At the attached project, there is a single Activity and a single xml layout with the map. No code anywhere to hold any reference of the map. The only change I did (to the generated project by
dependency to google-play-services_lib.

I also attach a heap dump after about 10 iterations.

PS. 85KB might seem little, but in my real app heap-usage increases by chunks of 500KB.

EclipseProject.zip
1.4 MB Download

leak.hprof
4.9 MB Download

fu...@gmail.com <fu...@gmail.com> #47

I have a memory issue related with Maps, it is a huge problem because my app makes an intensive use of maps. So please, I would appreciate if you can solve this issue and release a fix as

I've created a simple project that reproduce the problem.
It has just two activities:
- MainActivity: has a map and a button that starts second activity (with the intent flag FLAG_ACTIVITY_REORDER_TO_FRONT set).
- SecondActivity: has just a button that start the first activity (with the intent flag FLAG_ACTIVITY_REORDER_TO_FRONT set).

Every time that I navigate from SecondActivity to MainAcitivity(the one with the map). The heap increases and never goes down. Eventually the app crashes with an Out Of Memory exception

I have been able to reproduce the error with an HTC Desire with android 2.3.7 and with other 2.x phones. For the moment there is no problem with my Jiayu G2 (ICS 4.0.4).

I can only attach the eclipse project to this comment, but below you have the links to the project and the .hprof file.

- The eclipse project:
http://fjmontiel-beta.googlecode.com/files/MapLeaking.zip

- The .hprof file showing the leak;
http://fjmontiel-beta.googlecode.com/files/mapleaking.hprof

- You can also download the project from the repository (SVN):
http://fjmontiel-beta.googlecode.com/

MapLeaking.zip
2.8 MB Download

ju...@gmail.com <ju...@gmail.com> #48

What I did to manage it is to nullify the map object when leaving the activity so it is ready for garbage collection. Recreate the map or re-instantiate the map when going back to the map via
consumption at a minimum. Works for me right now on older phones.

fu...@gmail.com <fu...@gmail.com> #49

For me there is no improvement nullifying the map, the leak is still present. I've also tried detaching and attaching the fragment that contains the map but the problem persist.

[Deleted User] <[Deleted User]> #50

We've seen a leak with play services r6 when there's an editable field in the view hierarchy and the activity is destroyed/rebuilt. The InputMethodManager service accumulates maps.(obfusca
mText.mSpans member.

2.3.x devices are working fine with no problematic native or Dalvik heap growth, and removing editable fields or not marking non-editable fields selectable avoids leaking maps objects.

The test device is running 4.2.2 @JDQ39.

NB: OOMError is thrown and logged but a crash message may not appear.

Steps to reproduce:
1. Build an activity with an editable or selectable field.
2. Attach a fragment to the activity that contains a gms.maps.MapView.
3. Repeatedly destroy the activity or fragment.

Eventually, the Dalvik heap will be exhausted and the process will be terminated and restarted.

All lifecycle messages are being passed i.a.w. the maps v2 for Android API notes.

android5114994318883346319.hprof.xz
7.6 MB Download

[Deleted User] <[Deleted User]> #51

Update: Further testing showed that navigating away from the activity leaking maps objects would eventually cause IMM to release its references and GC the maps. The above behavior is a

ma...@gmail.com <ma...@gmail.com> #52

The EditText / IMM leak is known for some time already and not really related to maps.

As a side note I suggest not to put EditText on the same Activity as map to minimize the leak.

fu...@gmail.com <fu...@gmail.com> #53

Some info about this bug from Android staff would be appreciated. ETA for a new release? Some way to workaround these memory leaks?
I'm working on an app that MUST be released this Friday and I now I have to release it with this terrible bug that makes the app crash (I suspect that my client won't be very happy). I'm trying
situtation is very frustrating. I can't understand how this bug has not been fixed yet.

um...@gmail.com <um...@gmail.com> #54

Hi all!
I notice the same bug (OutOfMemoryException) with Maps API v2 and HTC Wildfire S.
I have a lot of markers in my map (about 2500). After about a minute of surfing the map, it crashes...

vi...@gmail.com <vi...@gmail.com> #55

I use maps API v2 in two activities of my application and spent a few days of my life just with testing and figuring out why the application crashes because of memory leaks. There is always

Three/four occurrences of:

```
04-26 17:24:19.765: E/dalvikvm-heap(17106): 1048576-byte external allocation too large for this process.
04-26 17:24:19.765: E/GraphicsJNI(17106): VM won't let us allocate 1048576 bytes
```

And shortly after the OOM exception occurs and the application crashes:

```
04-27 14:47:57.742: E/AndroidRuntime(4462): FATAL EXCEPTION: GLThread 262
04-27 14:47:57.742: E/AndroidRuntime(4462): java.lang.OutOfMemoryError: bitmap size exceeds VM budget
04-27 14:47:57.742: E/AndroidRuntime(4462): at android.graphics.Bitmap.nativeCreate(Native Method)
04-27 14:47:57.742: E/AndroidRuntime(4462): at android.graphics.Bitmap.createBitmap(Bitmap.java:477)
04-27 14:47:57.742: E/AndroidRuntime(4462): at maps.s.h.a(Unknown Source)
04-27 14:47:57.742: E/AndroidRuntime(4462): at maps.cr.a.a(Unknown Source)
04-27 14:47:57.742: E/AndroidRuntime(4462): at maps.cr.a.a(Unknown Source)
04-27 14:47:57.742: E/AndroidRuntime(4462): at maps.cr.a.b(Unknown Source)
04-27 14:47:57.742: E/AndroidRuntime(4462): at maps.l.m.a(Unknown Source)
04-27 14:47:57.742: E/AndroidRuntime(4462): at maps.l.at.a(Unknown Source)
04-27 14:47:57.742: E/AndroidRuntime(4462): at maps.y.ab.a(Unknown Source)
04-27 14:47:57.742: E/AndroidRuntime(4462): at maps.af.v.a(Unknown Source)
04-27 14:47:57.742: E/AndroidRuntime(4462): at maps.af.v.a(Unknown Source)
04-27 14:47:57.742: E/AndroidRuntime(4462): at maps.af.v.a(Unknown Source)
04-27 14:47:57.742: E/AndroidRuntime(4462): at maps.p.q.m(Unknown Source)
04-27 14:47:57.742: E/AndroidRuntime(4462): at maps.p.q.run(Unknown Source)
```

I have tested it on many devices from various manufacturers on all OS versions beginning 2.3 and the result is always the same, the only difference is WHEN the crash occurs.

On devices with larger memory for application it is necessary to run the application dozens of times, on devices with low memory for application – such as Sony Ericsson Xperia Arc without
immediately.

It crashes even if there are no user markers in the map and with no screen rotation enabled (!)

On devices running 3.0 and higher this can be suppressed by setting largeHeap flag in the manifest, by that's really not a solution...

I would welcome fixing this bug ASAP as it makes the new maps API totally unusable and we are missing or business. The application is almost done, works nice, but...crashes because of th

I would switch back to API v1, but unfortunately that's not possible. So what should I do now? I am at dead end...

cy...@gmail.com <cy...@gmail.com> #56

#58: I don't know your code nor what you are trying to do but the only thing I can say is you are probably trying to allocate too much memory.

The message "VM won't let us allocate 1048576 bytes" basically means you are trying to allocate a Bitmap of 512x512px (sqrt(1048576 / 4) in Bitmap.Config.ARGB_8888) which is quite hug
it's quite obvious you run out of memory.

ma...@gmail.com <ma...@gmail.com> #57

Comment has been deleted.

Message last modified on May 7, 2021 10:36PM

vi...@gmail.com <vi...@gmail.com> #58

Yes, of-course, the application allocates too much memory, but I do not use such a large bitmaps at all and the layout is static, no image downloading on background, just the map and a few
source of memory leaks – just to isolate the source. Now there is only a map that remains...

It is probably the gmaps API v2 which allocates such a big memory chunks and leaks. As you can see, I am not alone with exactly the same problem – the same logcat error. I use SupportMa
and also when zooming by pinch-to-zoom gesture.

The OOM exceptions come from OpenGL thread – i.e. from SurfaceView which map use for rendering.

I would not even suspect the gmaps API v2 if there is nobody with such a problem, but it is mentioned on StackOverflow as a confirmed bug – which led me to this forum...

I am working on a simple test application to demonstrate this.

cb...@google.com <cb...@google.com> #59

To help us diagnose any potential memory leaks, could you please answer the questions outlined in #25?

- Does the memory keep rising by a similar amount every time you rotate the device, with the app ultimately crashing?
- Are you able to provide us with a hprof file so that we can investigate what is causing the memory leak on your device.

vi...@gmail.com <vi...@gmail.com> #60

Comment has been deleted.

Message last modified on May 7, 2021 10:37PM

mi...@sprachgewalt.de <mi...@sprachgewalt.de> #61

I added a Robotium test case to the MapLeaking example by #50 to reproduce the problem on my test devices, see included zip. I have the same problem with my app.

The zip includes a converted hprof file.

Steps to reproduce:
1. Replace API key
2. Run MapLeakingTest
3. Watch heap grow until out of memory

The growing heap occurs on the following test devices:
 HTC Desire running 2.3.7 (Oxygen mod)
 HTC Desire running 2.2 (Stock HTC ROM)

It does *NOT* happen on:
 Motorola Xoom running 4.0.4

On the Motorola Xoom the heap size stays at 8.5 MB (with 7.8 MB allocated), on the
HTC devices the heap grows and grows with each click.

I opened the included hprof file (from one run of the test, in the middle of the run) in MAT and the Leak Suspect reports says:

One instance of "maps.ca.a" loaded by "dalvik.system.PathClassLoader @
0x40556d48" occupies 10,336,528 (77.35%) bytes. The memory is accumulated in one
instance of "java.util.HashMap$HashMapEntry[]" loaded by "<system class
loader>".

Please advise and thanks for looking into this!

Mike

MapLeakingWithTestAndHprof.zip
9.6 MB Download

vi...@gmail.com <vi...@gmail.com> #62

I have tested the MapLeaking example by #50 on three devices:

1, Nexus 7 (Android 4.2.2) – no memory leaks
2, LG-P970 (Android 2.3.4), leaking, see the attached hprof
3, Sony Ericsson Arc - LT15i (Android 2.3.3), huge memory leaks, see the attached hprof

Vita

lg_p970_2_3_4.zip
1.3 MB Download

sony_ericsson_arc_2_3_3.zip
1.7 MB Download

vi...@gmail.com <vi...@gmail.com> #63

Hi all,

I have analyzed the issue in more detail and found the main reason of the OOM exception in my application.

Although the gmaps v2 API really leaks on Android 2.3.x, the leakage is not very significant – only a few kilobytes. It can kill the application after a really extensive usage, not after a few app s

The real problem is in fact that I use also HoloEverywhere library:

https://github.com/Prototik/HoloEverywhere

which does leak a lot! It is not such a big problem when using only simple layouts, but in conjunction with Google maps v2 API it is the ultimate application killer.

I have tested it on my LG-P970 (Android 2.3.4). After the first application start there is about 4 MB and 70 000 objects allocated in the VM heap, after only five cycles of back/start it raises to
OOM exception and app kill in a short time.

For those who would check it personally I created a sample project, it is in the public SVN repository on Assembla:

https://www.assembla.com/code/gmaps_v2_mem_leak/subversion/nodes

Vita

cb...@google.com <cb...@google.com> #64

Marked as fixed, reassigned to cb...@google.com.

A number of memory leaks were fixed in the most recent version of the Google Maps Android API.

If you continue to see memory leaks, please file a new issue with an hprof dump.

ks...@mail.ru <ks...@mail.ru> #65

[Comment deleted]