Graphics.drawImage() in Java is EXTREMELY slow on some computers yet much faster on others

Asked 15 years, 9 months ago Modified 7 years, 9 months ago

Viewed 19k times

16

I'm having a strange problem, basically in Java Graphics.drawImage() is extremely slow on some computers and faster on others. This isn't related to the computers power either, some weaker computers run it fine while some stronger ones seem to choke up at the drawImage call.

It may or may not be related to the width and height, I have a very, very large width and height defined (something like 5000 by 2500). I wouldn't think it's the issue except like I said it runs in real time speed on some computers and slower on others and doesn't seem to be tied to the computers relative power.

Both computers have the same version of Java, both use Vista. One has a 1.83ghz Core 2 Duo with 1gb RAM and onboard graphics (runs everything fine), the other has a 2.53 ghz core 2 duo with a 9600GS (latest nVidia drivers) and 4gb of RAM and it literally chugs on the drawImage call.

Any ideas?

edit: ok this is really wierd, I'm drawing the image to a window in Swing, now when I resize the window and make it really small the image gets scaled down too and it becomes small. Suddenly everything runs smoothly, when I scale it back up to the size it was before it's still running smoothly!

It also has multiple monitor issues, if I do the resize trick to make it run faster on one monitor then scroll it over to another monitor when more than half of the window is in the new monitor it starts chugging again. I have to resize the window again to small then back to its original size to get back the speed.

If I do the resize trick on one monitor, move it over to the other it of course chugs, but if I return it back to the original monitor on which I did the resize trick it works 100%

If I have two swing windows open (displaying the same image) they both run slow, but if I do the resize trick on one window they both start running smoothly (however this isn't always the case).

*when I say resize the window I mean make it as small as possible to the point the image can't actually be seen.

Could this be a bug in Java maybe?

java **graphics**

Share

Improve this question

Follow

edited Mar 18, 2009 at 14:40

asked Mar 18, 2009 at 12:31

meds

What kind of image is it, and what versions of the JDK are on the computers? – Eric Petroelje Mar 18, 2009 at 12:40

I think I am seeing the same problem. This is JDK 8 on XP (yeah I know) and a Hi-Color display. It takes a couple of seconds to render a BufferedImage containing a full screen shot - but only the second (!) time the component is drawn. Subsequent calls to drawImage are instant again. Maybe some conversion takes place internally? Quite confusing all in all, and not really acceptable that a simple color conversion takes that long (if it's the reason). – Stefan Reich Dec 7, 2016 at 16:35

7 Answers

Sorted by: Highest score (default)

23

Performance of writing an image to a screen is very much affected by the format in which the image is stored. If the format is the same as the screen memory wants then it can be very fast; if it is not then a conversion must be done, sometimes pixel by pixel, which is very slow.

If you have any control over how the image is stored, you should store it in a format that the screen is looking for. Here is some sample code:

```
    GraphicsEnvironment env =
GraphicsEnvironment.getLocalGraphicsEnvironment();
    GraphicsDevice device = env.getDefaultScreenDevice
    GraphicsConfiguration config = device.getDefaultCo
    BufferedImage buffy = config.createCompatibleImage
Transparency.TRANSLUCENT);
    Graphics g = buffy.getGraphics();
```

If you are going to draw the image many times it may be worth converting to a compatible format even if it came in some other format.

Drawing an image will also be slower if you are transforming it as you draw, which the 'resizing' part of your description makes me think you might be. Again, do the resize once (when the window is resized) and cache the resized and compatible image so that it can be redrawn quickly.

Share Improve this answer Follow

edited Mar 2, 2017 at 22:18

answered Mar 18, 2009 at 18:25

DJClayworth

26.8k ● 9 ● 58 ● 80

3 this worked brilliantly for me, thank you! I used this code alongside getRGB / setRGB to convert my BufferedImage – Sam Mar 6, 2011 at 11:09

Thank you so much for this! Went from ~17fps to 40 – René Jensen Dec 25, 2012 at 10:46

I see that you're creating a `BufferedImage` by using `createCompatibleImage`, but then what? That just gives you a blank image, right? How do you load up, say, a png file into that `BufferedImage`? – nullromo Feb 9, 2017 at 5:38

1 @nullromo I figured it out. You can simply use `Graphics g = buffy.getGraphics()` and then `g.draw~~~()` on that. – nullromo Feb 10, 2017 at 21:51

5

If you are using sun's Java try some of the following system properties, either as command line parameters or the first lines in main

```
sun.java2d.opengl=true  //force ogl
sun.java2d.ddscale=true //only when using direct3d
sun.java2d.translaccel=true //only when using
direct3d
```

more flags can be viewed at this page

Look at `sun.java2d.trace` which can allow you to determine the source of less-than-desirable graphics performance.

Share Improve this answer

Follow

edited Feb 28, 2011 at 16:17

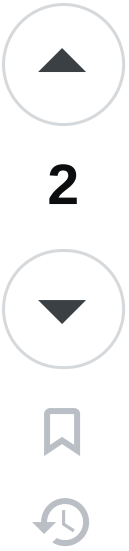
Joachim Sauer

308k ● 59 ● 565 ● 620

answered Mar 18, 2009 at 13:48

KitsuneYMG

12.9k ● 5 ● 38 ● 58



There are several things that could influence performance here:

- Available RAM
- CPU speed
- Graphic card (onboard or seperate)
- Graphic driver
- Java version
- Used video mode (resolution, bitdepth, acceleration support)

EDIT: Having a look at the edited question, I'd propose to check if the 9600GS system has the newest NVIDIA drivers installed. I recently installed a driver for an Intel onboard graphics card that replaced the generic Windows driver and made moving windows, watching videos, browsing etc. a lot faster.

All the other specs look good. Perhaps Java doesn't detect the 9600GS and doesn't use hardware acceleration, but I doubt this.

Also check the OS configuration. On Windows, you can turn off hardware acceleration for debugging purposes.

Of course the best way to handle this would be to change your code - resize the image or split it up into chunks as DNS proposed. You'll never be able to see the whole image as it is on the screen.

Share Improve this answer

Follow

edited Mar 18, 2009 at 13:02

answered Mar 18, 2009 at 12:38

schnaader

49.7k ● 10 ● 108 ● 139

> One computer is old, we're talking really old and the other is new and high end, it has better CPU, RAM, graphics but the same version of Java. Also I might have been wrong with the 50,000x25,000, it's actually 5000x2500. – meds Mar 18, 2009 at 12:43

2

How are you judging the computers' power? A 50x25 K 32-bit image takes more than 4.5 GB RAM to hold in memory (50000 * 25000 * 4 bytes). If one computer has more RAM than another, that can make a huge difference in speed, because it won't have to swap to disk as often. You should consider grabbing subsections of the image and working with those, instead of the whole thing.

Edit: Are you using the latest Java & graphics drivers? If your image is only 5Kx2.5K, the only thing I can think of is that it's doing it without any hardware acceleration.

Share Improve this answer

Follow

edited Mar 18, 2009 at 13:20

answered Mar 18, 2009 at 12:39

DNS
38.2k ● 20 ● 96 ● 133

no I meant 5000x2000, I did find the 50000x20000 number rather large then realized I was putting an extra 0 at the end of the numbers, I might be slowly going blind or something. – meds Mar 18, 2009 at 12:46

1

Check the screen settings. My bet is that pixel depth is different on the two systems, and that the slow one has an odd pixel depth related to the image object you are trying to display.

Share Improve this answer

Follow

answered Mar 18, 2009 at 12:59

Thorbjørn Ravn Andersen
75.3k ● 34 ● 199 ● 352

0

Since Java uses OpenGL to do 2D drawing, the performance of your app will be affected by the OpenGL performance of the graphics chip in the respective computer. Support for OpenGL is dwindling in the 3D industry, which means that (ironically) newer chips may be slower at OpenGL rendering than older ones - not only due to hardware but also drivers.

Share Improve this answer

Follow

answered Mar 18, 2009 at 13:02

Ian Kemp
29.8k ● 21 ● 120 ● 152

Have you tried Full-Screen Exclusive Mode?

This might help:
http://download.oracle.com/javase/tutorial/extra/fullscreen/index.html

0

Share Improve this answer

Follow

answered Feb 16, 2011 at 21:57

lasantha

855 ● 8 ● 7