darktable - Bug #10764

Slow hover response over Lighttable Thumbnails

12/09/2015 02:41 AM - John Morris

Status:	Duplicate	**Start date:**	12/09/2015
Priority:	Low	**Due date:**	
Assignee:	Pascal Obry	**% Done:**	0%
Category:	Lighttable	**Estimated time:**	0.00 hour
Target version:	2.6.1		
Affected Version:	2.0rc3	**bitness:**	64-bit
System:	Ubuntu	**hardware architecture:**	amd64/x86

Description

I noticed a performance problem in Lighttable mode where there is a slow response while moving the mouse over a series of thumbnails.
For instance, moving the pointer over a series of 10 tumbnails took more than one second for the mouse-over selection to finish updating the thumbnail status and Image Information for the track the mouse pointer took.
In other words, screen updates were not fast enough to keep up with the mouse pointer.

However, the 2.0~rc3+108~g4ae6a1b-0pmjdebruijn1~trusty repository update I installed today seems to have made an improvement although there is still a little lag. Maybe this is not a bug anymore, but I thought it best to report it.

History

#1 - 12/09/2015 02:43 PM - Tobias Ellinghaus

- Status changed from New to Incomplete

- % Done changed from 0 to 20

Please try disabling thumbnail color management in the 2nd tab of the preferences. That should speed it up a bit but might still be slow.

#2 - 12/09/2015 03:31 PM - John Morris

Tried disabling thumbnail color management but it didn't seem to make a noticeable difference. There is much more lag in mouse-over response than I recall in 1.6.x.

#3 - 01/02/2016 09:16 PM - Stig Inge Lea Bjørnsen

I'm experiencing a similar issue in Darktable 2.0 using Debian Sid. In the Lighttable, moving the mouse over the thumbnails or over the area below all the thumbnails generates a lot of CPU load.

When running Darktable with debug output, moving the mouse pointer one pixel as described above generates repeated output like this:

```
[lighttable] image expose took 0,0195 sec
```

followed by:

```
[mipmap_cache] thumbs fill 37,00/4096,00 MB (0,90%)
[mipmap_cache] float fill 0/8 slots (0,00%)
[mipmap_cache] full  fill 2/8 slots (25,00%)
[mipmap_cache] level | near match | miss | stand-in | fetches | total rq
[mipmap_cache] thumb |   0,36% |   0,18% | 100,00%  |   0,00% | 100,00%
[mipmap_cache] float |   -nan% |   -nan% |   0,00%  |   0,00% |   0,00%
[mipmap_cache] full  |   -nan% |   -nan% |   0,00%  | 100,00% |   0,00%

[lighttable] expose took 0,2182 sec
```

#4 - 05/16/2016 09:57 PM - Stig Inge Lea Bjørnsen

I have compiled from master@ca7497303b716f32e55fb9684ba4edeea1b8f6b2 using GTK 3.20 which has event compression enabled by default.

Both the lightroom view and the filmstrip at the bottom of the darkroom view are redrawn on every pixel the pointer is moved over them. This is the cause of the sluggishness.

Filmstrip

From src/libs/tools/filmstrip.c:

```
static gboolean _lib_filmstrip_motion_notify_callback(GtkWidget *w, GdkEventMotion *e, gpointer user_data)
{
  dt_lib_module_t *self = (dt_lib_module_t *)user_data;
  dt_lib_filmstrip_t *strip = (dt_lib_filmstrip_t *)self->data;

  strip->pointerx = e->x;
  strip->pointery = e->y;

  /* redraw */
  gtk_widget_queue_draw(self->widget);
  return TRUE;
}
```

Lighttable

Redrawing of the lighttable as the widget in the center is enqueued to `gtk_widget_queue_draw` by src/gui/gtk.c:

```
static void _ui_widget_redraw_callback(gpointer instance, GtkWidget *widget)
```

which is triggered by DT_SIGNAL_CONTROL_REDRAW_CENTER raised from src/views/lighttable.c:

```
void mouse_moved(dt_view_t *self, double x, double y, double pressure, int which)
{
  dt_control_queue_redraw_center();
}
```

which is called from src/views/view.c:

```
void dt_view_manager_mouse_moved(dt_view_manager_t *vm, double x, double y, double pressure, int which)
```

which is called from src/control/control.c:

```
void dt_control_mouse_moved(double x, double y, double pressure, int which)
```

which is called from gui/gtk.c:

```
static gboolean mouse_moved(GtkWidget *w, GdkEventMotion *event, gpointer user_data)
```

which is set to receive the motion-notify-event in gui/gtk.c:

```
int dt_gui_gtk_init(dt_gui_gtk_t *gui, int argc, char *argv[])
```

#5 - 03/30/2017 10:47 AM - Peter Grodovsky

Hello, was there any development regarding this problem? Can I help by providing some more info? It really seems that lighttable is being redrawn constantly during mouse movement. Just moving the mouse around (even whithin the same thumbnail) generates over 200% CPU load for the darktable process (quad-core Intel i5-4590S CPU + 16GB RAM). This is on DT 2.2.3.

The experience is highly dependent on the window size and the size/number of thumbnails. This unfortunately makes lighttable almost unusably slow on a 4K display. Example debug (constantly repeated while the mouse moves) with a full-screen window on a 4096x2160 resolution, 6 thumbnails visible:

```
[sql] /build/darktable-32pdEm/darktable-2.2.3/src/common/image.c:629, function dt_image_altered(): prepare "SE
LECT operation FROM main.history WHERE imgid = ?1"
[lighttable] image expose took 0,0394 sec
[sql] /build/darktable-32pdEm/darktable-2.2.3/src/common/image.c:629, function dt_image_altered(): prepare "SE
LECT operation FROM main.history WHERE imgid = ?1"
[lighttable] image expose took 0,0420 sec
[sql] /build/darktable-32pdEm/darktable-2.2.3/src/common/image.c:629, function dt_image_altered(): prepare "SE
LECT operation FROM main.history WHERE imgid = ?1"
[lighttable] image expose took 0,0397 sec
[sql] /build/darktable-32pdEm/darktable-2.2.3/src/common/image.c:629, function dt_image_altered(): prepare "SE
LECT operation FROM main.history WHERE imgid = ?1"
[lighttable] image expose took 0,0491 sec
[sql] /build/darktable-32pdEm/darktable-2.2.3/src/common/image.c:629, function dt_image_altered(): prepare "SE
LECT operation FROM main.history WHERE imgid = ?1"
[lighttable] image expose took 0,0122 sec
[sql] /build/darktable-32pdEm/darktable-2.2.3/src/common/image.c:629, function dt_image_altered(): prepare "SE
LECT operation FROM main.history WHERE imgid = ?1"
[lighttable] image expose took 0,0499 sec
[sql] /build/darktable-32pdEm/darktable-2.2.3/src/common/image.c:629, function dt_image_altered(): prepare "SE
LECT operation FROM main.history WHERE imgid = ?1"
[lighttable] image expose took 0,0072 sec
[sql] /build/darktable-32pdEm/darktable-2.2.3/src/common/image.c:629, function dt_image_altered(): prepare "SE
LECT operation FROM main.history WHERE imgid = ?1"
[lighttable] image expose took 0,0133 sec
[sql] /build/darktable-32pdEm/darktable-2.2.3/src/common/image.c:629, function dt_image_altered(): prepare "SE
LECT operation FROM main.history WHERE imgid = ?1"
[lighttable] image expose took 0,0148 sec
[mipmap_cache] thumbs fill 245,96/2048,00 MB (12,01%)
[mipmap_cache] float fill 0/4 slots (0,00%)
[mipmap_cache] full  fill 2/4 slots (50,00%)
[mipmap_cache] level | near match | miss | stand-in | fetches | total rq
[mipmap_cache] thumb |   0,35% |   0,10% | 100,00%  |  85,71% | 100,00%
[mipmap_cache] float |   -nan% |   -nan% |   0,00%  |   0,00% |   0,00%
[mipmap_cache] full  |   -nan% |   -nan% |   0,00%  |  14,29% |   0,00%

[lighttable] expose took 0,2704 sec
```

After resizing the window to roughly quarter of the screen (so about the size of standard full-hd display) the response is better, but the issue still persists:

```
[sql] /build/darktable-32pdEm/darktable-2.2.3/src/common/image.c:629, function dt_image_altered(): prepare "SE
LECT operation FROM main.history WHERE imgid = ?1"
[lighttable] image expose took 0,0079 sec
[sql] /build/darktable-32pdEm/darktable-2.2.3/src/common/image.c:629, function dt_image_altered(): prepare "SE
LECT operation FROM main.history WHERE imgid = ?1"
[lighttable] image expose took 0,0078 sec
[sql] /build/darktable-32pdEm/darktable-2.2.3/src/common/image.c:629, function dt_image_altered(): prepare "SE
LECT operation FROM main.history WHERE imgid = ?1"
[lighttable] image expose took 0,0081 sec
[sql] /build/darktable-32pdEm/darktable-2.2.3/src/common/image.c:629, function dt_image_altered(): prepare "SE
LECT operation FROM main.history WHERE imgid = ?1"
[lighttable] image expose took 0,0090 sec
[sql] /build/darktable-32pdEm/darktable-2.2.3/src/common/image.c:629, function dt_image_altered(): prepare "SE
LECT operation FROM main.history WHERE imgid = ?1"
[lighttable] image expose took 0,0050 sec
[sql] /build/darktable-32pdEm/darktable-2.2.3/src/common/image.c:629, function dt_image_altered(): prepare "SE
LECT operation FROM main.history WHERE imgid = ?1"
[lighttable] image expose took 0,0077 sec
```

```
[sql] /build/darktable-32pdEm/darktable-2.2.3/src/common/image.c:629, function dt_image_altered(): prepare "SE
LECT operation FROM main.history WHERE imgid = ?1"
[lighttable] image expose took 0,0032 sec
[sql] /build/darktable-32pdEm/darktable-2.2.3/src/common/image.c:629, function dt_image_altered(): prepare "SE
LECT operation FROM main.history WHERE imgid = ?1"
[lighttable] image expose took 0,0056 sec
[sql] /build/darktable-32pdEm/darktable-2.2.3/src/common/image.c:629, function dt_image_altered(): prepare "SE
LECT operation FROM main.history WHERE imgid = ?1"
[lighttable] image expose took 0,0063 sec
[mipmap_cache] thumbs fill 245,96/2048,00 MB (12,01%)
[mipmap_cache] float fill 0/4 slots (0,00%)
[mipmap_cache] full  fill 2/4 slots (50,00%)
[mipmap_cache] level | near match | miss | stand-in | fetches | total rq
[mipmap_cache] thumb |   0,35% |   0,11% | 100,00%  |  85,71% | 100,00%
[mipmap_cache] float |   -nan% |   -nan% |   0,00%  |   0,00% |   0,00%
[mipmap_cache] full  |   -nan% |   -nan% |   0,00%  |  14,29% |   0,00%

[lighttable] expose took 0,0609 sec
```

It's worth noting that the response is very slow also when navigating with keyboard, not touching the mouse (and it depends on the window size as well). So maybe the constant redraw is not the only problem.

Thanks!

#6 - 08/03/2017 01:11 PM - Paolo Astengo

Is there any news about this ticket?

#7 - 12/27/2017 05:07 PM - Peter Grodovsky

Unfortunately the issue still persists in new darktable 2.4.0 - mouse navigation in lighttable consumes most of the CPU and is very lagging on a HiDPI display.

#8 - 02/13/2018 06:15 AM - [redacted]

Peter Grodovsky wrote:

> Unfortunately the issue still persists in new darktable 2.4.0 - mouse navigation in lighttable consumes most of the CPU and is very lagging on a HiDPI display.

Yup, I just ran into this when I bought a new laptop. As a workaround, you can reduce your resolution, which should mitigate the issue.

As for an actual fix, what needs to be done and how can I help?

#9 - 07/12/2018 11:32 AM - Peter Schneider

I see exactly the same issue on my Dell XPS 15 using the built-in 4K display.

Although using OpenCL on quite a powerful GTX 1050 GPU, I do net see much of a difference in performance during editing or culling the images in lighttable compared to my 5 years older x220 without a GPU but with a 1366x768 resolution.
I'd be happy to contributing to fixing this issue as I fear this may hinder darktable's adoption on new systems due to missing compatibility.
Any suggestions where we could start?

#10 - 08/11/2018 01:37 PM - Nils Holle

Ran into this issue as well, I bought a 4K display especially for photo editing and was quite disappointed.
I haven't done any Darktable development yet, but would be happy to contribute too.

I took a (very) quick look at the code, the relevant method taking quite a lot of time is dt_view_image_expose in view.c, which is like 600 lines long and I don't get a lot of what happens.
I would suggest profiling that to see which parts are critical, but perhaps there already is some knowledge about that?

If I observe correctly the thumbnail image seems to be redrawn (or "exposed") on every mouse movement, which is extremely unnecessary.

#11 - 08/11/2018 01:38 PM - Nils Holle

I would also suggest to increase the priority of this issue to medium as more and more users will use Hidpi displays (especially the Mac users).

#12 - 08/14/2018 05:56 PM - Andreas Schneider

Nils, increasing the priority will not help. darktable is a spare time projects and someone needs to find the spare time to work on this issue. If you want to take this task, I suggest you subscribe to the devel mailing list and if you have questions about the code ask there. This code probably needs a rewrite, not just some fixing.

#13 - 08/19/2018 08:09 PM - Pavol Stugel

Same problem on Windows 10 (GTX 750 TI) with higher resolution than 1080 (1080 works great, but 1440p unusable ...).

Full window repaint when moving with mouse (1px - 1x repaint, 100px - 100x repaint) ... This is problem even on 1080 but "lag" is not visible ...

#14 - 08/20/2018 05:31 PM - Rui Carmo

Just to add another datapoint, I use DT on Mac, Windows and Linux and have very little trouble with DT on Windows (Surface, HIDPI display) but find DT unbearably slow on a 27" Retina iMac and a Retina MacBook Pro.

I do feel around half a second lag, but not just inside the lighttable - most UI elements and menus feel sluggish as well (or at least more sluggish on the Macs than on the other machines).

#15 - 12/10/2018 02:47 PM - John Morris

Unfortunately this problem still exists in 2.6.0rc1, especially on my 5K Retina 2015 iMac. Makes navigating around different images pretty much unusable.

#16 - 01/04/2019 05:00 PM - Ivan Evtukhovich

DT unusable on 4k display on MacOS :-(

./darktable --version
this is darktable 2.6.0
copyright (c) 2009-2018 johannes hanika
darktable-dev@lists.darktable.org

compile options:
bit depth is 64 bit
normal build
SSE2 optimized codepath enabled
OpenMP support enabled
OpenCL support enabled
Lua support enabled, API version 5.0.0
Colord support disabled
gPhoto2 support enabled
GraphicsMagick support enabled
OpenEXR support enabled

#17 - 01/05/2019 01:31 PM - Pascal Obry

If someone can test this:
https://github.com/darktable-org/darktable/pull/1973

#18 - 01/05/2019 03:32 PM - Nils Holle

- File darktable-9.mp4 added

- File darktable.mp4 added

- File darktable-4.mp4 added

I think it didn't really get much better, maybe a tiny bit (current faster-lighttable-2 branch). It is still quite horrible to use...
Interestingly, there seems to be some kind of sweet spot at nine images in a row (on the faster-lighttable-2 branch as well as the master branch).
Kubuntu 18.04 with Core i5-3570 and Nvidia GTX 1050 Ti

#19 - 01/05/2019 03:52 PM - Pascal Obry

Are the three videos done with dt from the branch faster-lighttable-2?

#20 - 01/05/2019 03:53 PM - Nils Holle

Yes, they are. 3840x2160 resolution, KDE 1.6 scaling.

Edit: In fullscreen mode

#21 - 01/05/2019 03:59 PM - Paolo Astengo

I'm not sure, but I guess it can depend on calling SQL functions for *every* mouse move. If the sql could be called only once per image, when the mouse intercepts the square where the image is, probably it can be possible to spare a lot of CPU power and time. Currently, a test for a change on the image under the cursor is called continuously.

#22 - 01/05/2019 04:03 PM - Pascal Obry

No, that's the point of my patch above. I have filtered out most mouse_move() events and this avoid lot of SQL calls and lot of cairo redraw (only the lighttable filemanager mode for now).

#23 - 01/05/2019 04:15 PM - Pascal Obry

@Nils, could you uncomment in lighttable.c the lines:

```
printf("evt (%.2f x %.2f) total %4d - skip %4d - done %4d\n     (thumb size %d, y_offet %d)\n",
           x, y, lib->etot, lib->eskip, lib->etot - lib->eskip, lib->thumb_size, y_offset);
```

And report the last 2 lines displayed in the console?

It would be interesting to see in your context how many events have been filtered out.

#24 - 01/05/2019 04:37 PM - Nils Holle

Tried a few different thumb sizes:

evt (2135.79 x 1510.23) total 79 - skip 68 - done 11
(thumb size 683, y_offet 144)

evt (2193.00 x 1198.07) total 80 - skip 68 - done 12
(thumb size 911, y_offet 287)

evt (1565.67 x 1045.79) total 185 - skip 140 - done 45
(thumb size 390, y_offet 265)

evt (1473.37 x 1051.96) total 186 - skip 140 - done 46
(thumb size 341, y_offet 28)

#25 - 01/05/2019 04:41 PM - Nils Holle

Just to add another setup: Ubuntu 18.10 on a Core m3-5Y10, no OpenCL, 3200x1800, Gnome 2x scaling: Same problems on faster-lighttable-2 branch.

#26 - 01/05/2019 04:45 PM - Nils Holle

I think filtering out events is a good idea to reduce the overall CPU usage; however, the actual problem is that a single lighttable expose takes far too long on high dpi screens, as all images are redrawn (if I got that right).

#27 - 01/05/2019 04:52 PM - Pascal Obry

Not not all images. At least that's not what is done on the code. Or I may be missing something.

#28 - 01/05/2019 05:02 PM - Pascal Obry

Oh! You're right we do redraw all thumbs each time we are moving...

#29 - 01/05/2019 05:02 PM - Nils Holle

I'm not quite sure... However, there seems to be some bottleneck in expose, and unfortunately an event filtering won't help in that case, I think.

Nevertheless, I think all users will profit from less CPU usage and longer battery lifetimes.

#30 - 01/05/2019 05:57 PM - Pascal Obry

@Nils, another attempt to filter out events and expose of the thumbs. Can you test again? That's still a WIP and I do expect to selection issues.

#31 - 01/05/2019 09:19 PM - Nils Holle

- File scrolling-8.mp4 added

- File scrolling-9.mp4 added

Much, much better, thank you so much!
There is still some room for improvement though; the scrolling is very painful (see videos attached).
Again, there's a very strange sweet spot at nine images in a row; at eight images, it's barely usable.

#32 - 01/05/2019 10:11 PM - Pascal Obry

There is no way around the scrolling slowness I think (we'll need a whole redesign of the lighttable, is this is far much work, my goal is to make the lighttable "usable" for 2.8).

What do you mean by a "sweet spot"? Sorry I'm not a native English speaker :)

#33 - 01/05/2019 11:01 PM - Nils Holle

Ok I get that, this is already much better than before. By sweet spot I mean that it works really well with nine images in a row, but not too well for all other zoom settings.

#34 - 01/05/2019 11:02 PM - Nils Holle

Is there any chance of getting this fix into the 2.6.1 release?

#35 - 01/06/2019 06:41 PM - Pascal Obry

I see no reasons for the sweet spot then :(

For 2.6.1, maybe will see but then this needs a quite extensive testing phase. Even if the patch is not that big, it is quite delicate and in a part that we cannot break (think about all filtering settings, the stars, local copies...). So please do as much testing as possible (and any dev around). It will be easier when merged to master, and report issues preferably in GitHub. Thanks.

#36 - 01/06/2019 06:41 PM - Pascal Obry

- Assignee set to Pascal Obry

- Status changed from Incomplete to In Progress

- % Done changed from 20 to 50

#37 - 01/06/2019 10:35 PM - Pascal Obry

- Target version set to 2.6.1

- Status changed from In Progress to Duplicate

- % Done changed from 50 to 0

I'm closing this as the discussion has moved here:
https://github.com/darktable-org/darktable/pull/1973

Files

darktable-4.mp4	1.15 MB	01/05/2019	Nils Holle
darktable.mp4	1.25 MB	01/05/2019	Nils Holle
darktable-9.mp4	1.02 MB	01/05/2019	Nils Holle
scrolling-8.mp4	3.35 MB	01/05/2019	Nils Holle
scrolling-9.mp4	5.48 MB	01/05/2019	Nils Holle