darktable - Bug #11872

local contrast (local laplacian) display bug when changing image

12/21/2017 07:54 PM - Matthieu Moy

Status:	Fixed	**Start date:**	12/21/2017
Priority:	Low	**Due date:**	
Assignee:		**% Done:**	100%
Category:	Darkroom	**Estimated time:**	0.00 hour
Target version:	2.6.0		
Affected Version:	git master branch	**bitness:**	64-bit
System:	Ubuntu	**hardware architecture:**	amd64/x86

Description

In darkroom, with the local contrast tool (new default mode: local laplacian), when switching from an image to another, I get a strange display bug: it seems darktable computes a mask for an image, and re-uses the mask when displaying the next image.

See example attached. Screenshots done with two images (image 1 and image 2), switching back and forth between the images.

The display bug disappears as soon as I move or zoom the image (i.e. when a new computation is triggered).

Thanks in advance,

Associated revisions

Revision bfe9911a - 11/01/2018 12:57 PM - Matthieu Moy

bilat: don't grab data from preview pipe if hash==0, fixes #11872

When changing image in darkroom, and when both the previous and the new image have local laplacian activated, the optimization grabing data from the preview pixelpipe introduced in 911133c (local laplacian: make roi aware in darkroom mode, 2017-10-15) was actually grabing data from the previously selected image.

The guilty line was:

```
if(hash != 0 && !dt_dev_sync_...)
```

in case hash == 0, the sync is not done, but the else branch still does the grabbing. Change this to exhibit 3 cases: 1) don't try to sync and grab, 2) sync failure, 3) sync success.

History

#1 - 08/08/2018 12:57 PM - Matthieu Moy

Note: this happens only when OpenCL is disabled.

(An expensive workaround is therefore to buy a good GPU and activate OpenCL ;-))

#2 - 08/08/2018 01:52 PM - Matthieu Moy

I found the guilty code:

https://github.com/darktable-org/darktable/blob/1ddd9bde8aa65ef321b7e7882b5e345b8d2cb694/src/iop/bilat.c#L296-L315

Introduced by: https://github.com/darktable-org/darktable/commit/911133c87b45be22251cf7252ed729d1632a27b2

If I replace the if (... < 0.9) by if (... < 0.0) (always false), then the issue disappears. What happens is that the full pipeline steals data from the preview pipeline, and apparently it steals it from the previously computed image.

According to gdb, the visual bug happens when hash is 0, i.e. when the if (hash != 0 && dt_dev_sync...) condition is false because of the left hand side of the &&. Then, the pipeline sync is not done, but the stealing from the preview pipeline still happens (else branch).

I don't fully understand the logic here, but I'll try a naive fix (at least it should be simpler for a core dev to finish debugging/fixing with this initial attempt).

#3 - 11/01/2018 01:29 PM - Anonymous

- % Done changed from 0 to 100

- Status changed from New to Fixed

Applied in changeset darktable|bfe9911ad0ca54e76b6faa3a036f68d56a11db13.

#4 - 11/29/2018 02:13 PM - Roman Lebedev

- Target version set to 2.6.0

Files

image-1-ok.jpg	99.7 KB	12/21/2017	Matthieu Moy
image-2-ko.jpg	91.7 KB	12/21/2017	Matthieu Moy
image-1-ko.jpg	101 KB	12/21/2017	Matthieu Moy