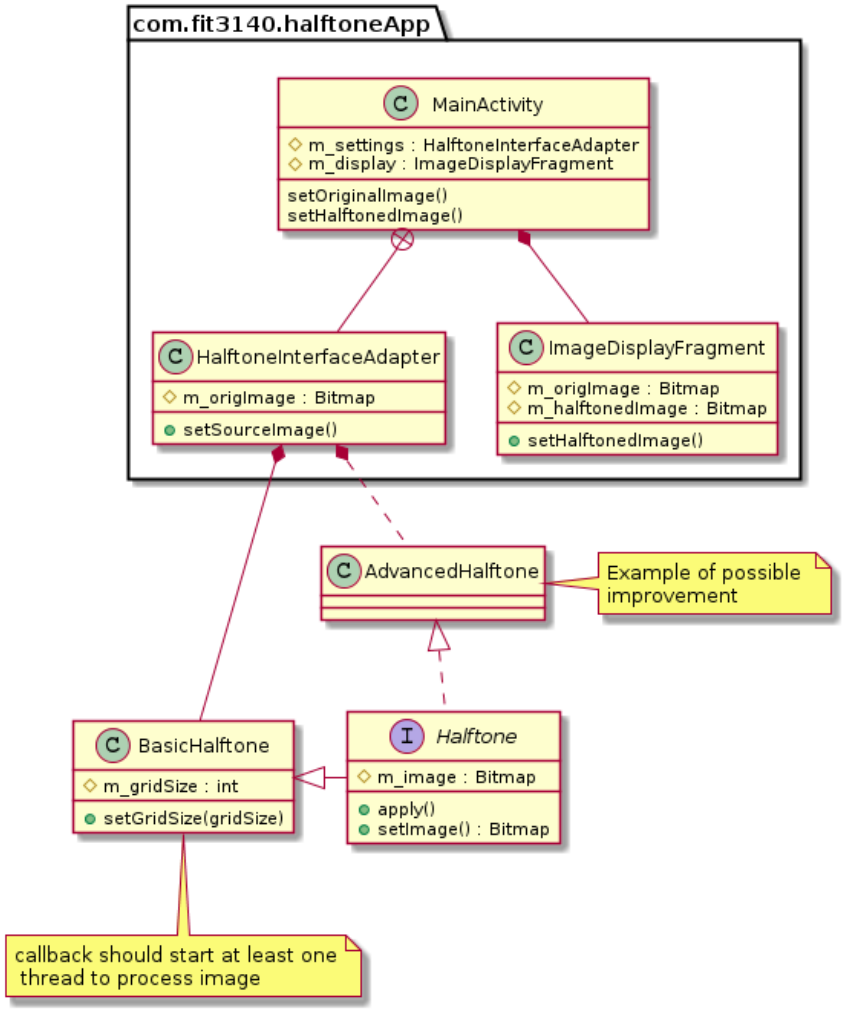
uml

Jack Hosemans

April 27, 2014

Contents

1 UML diagram



2 Descriptions of classes

2.1 MainActivity

This class should just be a joiner class for all the fragments.

2.1.1 setOriginalImage

This should only be called by the ImageDisplayFragment when the source image changes.

It changes the reference to the `m_origImage` value to be a reference to the image in the ImageDisplayFragment so that it can be easily be passed on to be halftoned.

2.1.2 setHalftonedImage

This should call the `setHalftonedImage()` method in the `ImageDisplayFragment` class so that a resultant halftoned image can be displayed to the user.

2.2 ImageDisplayFragment

This fragment should show the user images before and after halftoning.

Right now it does not matter how it is done, but something nice should be made so that swapping back to the original image should be painless and automatic when settings are changed.

2.2.1 setHalftonedImage

Set the displayed image to the halftoned image. Should also bring the image to the forefront so that the user can see the results.

2.3 HalftoneInterfaceAdapter

Note that this is an inner class of `MainActivity`.

This is what displays the settings for the current halftoning mode, mostly for encapsulation within the `MainActivity` so that no "special case" code has to be implemented in it.

Should implement a `ViewPager` so that the user can swipe between the halftoning modes that are implemented. Hopefully there will be more than one otherwise this will be wasted.

Each element in the `ViewPager` should be a fragment that implements its own interface to the user.

Should also have an "Apply" button for the user to press, to tell the currently selected halftone implementation to start processing.

2.3.1 setImage

Sets the reference image to be passed on to the halftone implementation when the user presses "Apply"

2.4 Halftone

This is an interface that all halftoning methods should implement. Basic stuff.

2.4.1 setImage

Called when the

2.5 BasicHalftone

This is the meat of the app, applys the halftoning that we all know and love(?).

Should be able so set the grid size, there really shouldn't be much more than that in this implementation.

Side note:

2.5.1 Proposal:

Halftoning usually takes a long time (>10s for large images.) A solution should be made for this.

- multiple threads could be used to work around the slowness of the application.
  - This would give the advantage of not slowing down the GUI
  - However, it would be harder to code. Maybe another spike is in order for implementation?
- A hard limit on the grid size would also be sufficient
  - "worse" than the threaded way because the application would appear to freeze when being halftoned. This is VERY bad.
  - Worse output due to each dot being bigger than any other way.
  - Not using the full power of the processor (We need all we can get)

2.6 AdvancedHalftone

This is currently just a placeholder in case we extend the application to do more than basic halftoning.

Implementations:

- Different colors
- Different angles
- dithering?