Class java.awt.image.RGBImageFilter

```
java.lang.Object
   |
   +----java.awt.image.ImageFilter
           |
           +----java.awt.image.RGBImageFilter
```

---

public class **RGBImageFilter**
extends ImageFilter

This class provides an easy way to create a ImageFilter which modifies the pixels of an image in the default RGB ColorModel. It is meant to be used in conjunction with a FilteredImageSource object to produce filtered versions of existing images. It is an abstract class that provides the calls needed to channel all of the pixel data through a single method which converts pixels one at a time in the default RGB ColorModel regardless of the ColorModel being used by the ImageProducer. The only method which needs to be defined to create a useable image filter is the filterRGB method. Here is an example of a definition of a filter which swaps the red and blue components of an image:

```
class RedBlueSwapFilter extends RGBImageFilter {
    public RedBlueSwapFilter() {
        // The filter's operation does not depend on the
        // pixel's location, so IndexColorModels can be
        // filtered directly.
        canFilterIndexColorModel = true;
    }
    public int filterRGB(int x, int y, int rgb) {
        return ((rgb & 0xff00ff00)
                | ((rgb & 0xff0000) >> 16)
                | ((rgb & 0xff) << 16));
    }
}
```

See Also:
FilteredImageSource, ImageFilter, getRGBdefault
Version:
1.8 09/08/95
Author:
Jim Graham

Variable Index

o **canFilterIndexColorModel**
This boolean indicates whether or not it is acceptable to apply the color filtering of the filterRGB method to the color table entries of an IndexColorModel object in lieu of pixel by pixel filtering.
o **newmodel**
o **origmodel**

Constructor Index

o **RGBImageFilter**()

Method Index

o **filterIndexColorModel**(IndexColorModel)
Filter an IndexColorModel object by running each entry in its color tables through the filterRGB function that RGBImageFilter subclasses must provide.
o **filterRGB**(int, int, int)
Subclasses must specify a method to convert a single input pixel in the default RGB ColorModel to a single output pixel.
o **filterRGBPixels**(int, int, int, int, int[], int, int)
Filter a buffer of pixels in the default RGB ColorModel by passing them one by one through the filterRGB method.
o **setColorModel**(ColorModel)
If the ColorModel is an IndexColorModel, and the subclass has set the canFilterIndexColorModel flag to true, then we substitute a filtered version of the color model here and whenever we see that original ColorModel object in the setPixels methods, otherwise we override the default ColorModel used by the ImageProducer and specify the default RGB ColorModel instead.
o **setPixels**(int, int, int, int, ColorModel, byte[], int, int)
If the ColorModel object is the same one that has already been converted, then simply pass the pixels through with the converted ColorModel, otherwise convert the buffer of byte pixels to the default RGB ColorModel and pass the converted buffer to the filterRGBPixels method to be converted one by one.
o **setPixels**(int, int, int, int, ColorModel, int[], int, int)
If the ColorModel object is the same one that has already been converted, then simply pass the pixels through with the converted ColorModel, otherwise convert the buffer of integer pixels to the default RGB ColorModel and pass the converted buffer to the filterRGBPixels method to be converted one by one.
o **substituteColorModel**(ColorModel, ColorModel)
Register two ColorModel objects for substitution.

Variables

o **origmodel**

```
protected ColorModel origmodel
```

o **newmodel**

```
protected ColorModel newmodel
```

o **canFilterIndexColorModel**

```
protected boolean canFilterIndexColorModel
```

This boolean indicates whether or not it is acceptable to apply the color filtering of the filterRGB method to the color table entries of an IndexColorModel object in lieu of pixel by pixel filtering. Subclasses should set this variable to true in their constructor if their filterRGB method does not depend on the coordinate of the pixel being filtered.

See Also:

substituteColorModel, filterRGB, IndexColorModel

Constructors

o **RGBImageFilter**

```
public RGBImageFilter()
```

Methods

o **setColorModel**

```
public void setColorModel(ColorModel model)
```

If the ColorModel is an IndexColorModel, and the subclass has set the canFilterIndexColorModel flag to true, then we substitute a filtered version of the color model here and whenever we see that original ColorModel object in the setPixels methods, otherwise we override the default ColorModel used by the ImageProducer and specify the default RGB ColorModel instead.

Overrides:

setColorModel in class ImageFilter

See Also:

ImageConsumer, getRGBdefault

o **substituteColorModel**

```
public void substituteColorModel(ColorModel oldcm,
                                 ColorModel newcm)
```

Register two ColorModel objects for substitution. If the oldcm is encountered during any of the setPixels methods, then the newcm will be substituted for it and the pixels will be passed through untouched (but with the new ColorModel object).

Parameters:

oldcm – the ColorModel object to be replaced on the fly
newcm – the ColorModel object to replace oldcm on the fly

o **filterIndexColorModel**

```
public IndexColorModel filterIndexColorModel(IndexColorModel icm)
```

Filter an IndexColorModel object by running each entry in its color tables through the filterRGB function that RGBImageFilter subclasses must provide. Use coordinates of –1 to indicate that a color table entry is being filtered rather than an actual pixel value.

Parameters:

icm – the IndexColorModel object to be filtered

Returns:

a new IndexColorModel representing the filtered colors

o **filterRGBPixels**

```
public void filterRGBPixels(int x,
                            int y,
                            int w,
                            int h,
                            int pixels[],
                            int off,
                            int scansize)
```

Filter a buffer of pixels in the default RGB ColorModel by passing them one by one through the filterRGB method.

See Also:

getRGBdefault, filterRGB

o **setPixels**

```
public void setPixels(int x,
                      int y,
                      int w,
                      int h,
                      ColorModel model,
                      byte pixels[],
                      int off,
                      int scansize)
```

If the ColorModel object is the same one that has already been converted, then simply pass the pixels through with the converted ColorModel, otherwise convert the buffer of byte pixels to the default RGB ColorModel and pass the converted buffer to the filterRGBPixels method to be converted one by one.

Overrides:

setPixels in class ImageFilter

See Also:
getRGBdefault, filterRGBPixels

o **setPixels**

```
public void setPixels(int x,
                      int y,
                      int w,
                      int h,
                      ColorModel model,
                      int pixels[],
                      int off,
                      int scansize)
```

If the ColorModel object is the same one that has already been converted, then simply pass the pixels through with the converted ColorModel, otherwise convert the buffer of integer pixels to the default RGB ColorModel and pass the converted buffer to the filterRGBPixels method to be converted one by one. Convert a buffer of integer pixels to the default RGB ColorModel and pass the converted buffer to the filterRGBPixels method.
Overrides:
setPixels in class ImageFilter
See Also:
getRGBdefault, filterRGBPixels

o **filterRGB**

```
public abstract int filterRGB(int x,
                              int y,
                              int rgb)
```

Subclasses must specify a method to convert a single input pixel in the default RGB ColorModel to a single output pixel.
See Also:
getRGBdefault, filterRGBPixels