Basic Rundown of Filters

Filters define effects to edit images without the need to make new copies of those images. These typically involve altering the colors of an image, but can also include effects like flipping the image horizontally. Filters used by the tool are kept in "filters.txt", and the tool will update "filter_effects.rpy" whenever it is run. The functions in filter_effects.rpy can be used on any image, but they must be kept in filters.txt for the tool to find and apply them to the image definition lines it creates.

To apply a filter to a character image, add the name of the filter to the tag you write in the show statement or dialogue abbreviation. You can apply any number of filters to a single pose. **This works both with poses defined by the tool and those not defined by the tool.**

Example:

```
show sayori u222141 dawn flip at t11
s "Yay! The sun's coming up!"
s 1k dawn "Wait, how'd my bow get on the other side?"
```

In the first line, the tool will apply the "flip" filter first, then the dawn filter to that image. (You can reverse this order in settings.txt if you prefer to apply filters in the order you type them.) The bottom row will also apply the dawn filter even though the pose isn't one the tool creates.

Applying Filters to a Full Scene?

Ren'Py is not able to take a filter and apply it to the entire screen at once. Instead the filter needs to be applied to each sprite being shown and the background separately. The tool has no problem applying filters to sprites for you as long as you name the ones you need in each line you write. You'll need to apply filters to backgrounds by hand, which is much simpler by comparison.

Each filter creates a Python function with the same name as the filter, which is what you'll be applying to your background. So if you need to apply the sepia filter to the clubroom, you can do so like this:

```
image bg club_day = "bg/club.png"
image bg club_day sepia = sepia("bg/club.png")
```

Existing Filters *(November 18, 2018)*

Here's a list of the filters initially released with the tool. (This list may be up to date or out of date in future versions—always check filters.txt for the most up to date list!)

sepia
Changes the image to sepia tone. Good for flashbacks, photographs, dreams, and ominous moments where the words "To Be Continued" scroll in from the right.

bw, gray
bw is a simple "black and white" filter. Gray takes an optional second argument from 0.0 to 1.0 indicating what percentage of the image's color should be left. Applying gray to a character pose is the same as applying bw to it—you'll need to define your lines by hand if you wish to specify a number.

ghost
Turn the image to a ghostly blue hue and make it slightly transparent. This takes an alpha value as the optional second argument—if you need to apply this to a background for any reason, it is recommend to use 1.0 for this argument (backgrounds probably shouldn't be see-through).

```
image bg club_day ghost = ghost("bg/club.png", 1.0)
```

dawn, morning, sunset, evening, night
Filters used in Monika Before Story for different times of day. These are usually perfectly acceptable for character sprites, but they can be hit or miss for backgrounds. Hand creating a new background for the time of day is usually the better result, but this may be an acceptable low-effort approach.

shadow, white
Shadow will turn the image entirely black, and white will turn the image entirely white. These can be used for character silhouettes or flash effects. Not recommended for backgrounds!

cloudy
Adjusts an image to look like a cloudy day, usually just done by desaturating the image slightly (a step closer to black and white). This takes "True" or "False" as an optional second argument to try and turn blue in the image to gray. You can try this on a background with a blue sky to make it seem more overcast. Once again, creating a background specifically to have a cloudy sky will usually give a more pleasant result.

flip
Horizontally flip the image. When used with character sprites, sometimes this looks just right, and sometimes it will cause obvious issues. For instance, Sayori's bow will be on the wrong side, and a heterochromatic character will suddenly have her eyes switch colors.

Adding New Filters

The filters in filters.txt are basically written in Python code, except that starting with "def" for the function names is optional so the file is easier to read, and the tool adds white space automatically. (You may include white space in the file, but the tool will remove it and add its own back in at run time.) A new filter should be written as a function whose first parameter is called "image" and has only optional parameters after that (if any). They should return an image in some fashion at the end, though the tool only really checks that there's a return line at all to recognize the filter function.

Most filter functions will simply be one line using an image manipulator function on the provided image and returning the result. They can be as complicated as need be, with cloudy being a good example of a more complex filter. The tool handles indentation by watching for lines which end with colons (start of a new block) and blank lines (which it takes as a signal that the most recent block just ended).

When the tool applies filters to character images, it provides the image as the only argument. This is why any additional parameters need to have default values—otherwise the tool won't provide anything and you'll get an error. Additional parameters are generally intended for backgrounds and other images you'll be applying the filters to.

The auto-generated file with all filters in it will alphabetize the filters, list them in a comment at the top, and add a function at the bottom named getCharacterImage. This function is courtesy of the Monika After Story dev team and is used by the tool for applying filters to any image, even poses it did not create. This most likely won't overlap with other functions you are using, but if it does, keep in mind that this one will be written into the file every time you run the tool.

For more information on what effects you can apply using filters, please refer to Ren'Py's documentation on Image Manipulators and on im.MatrixColor. Most filters are constructed using some combination of these functions.

If you have trouble, don't be afraid to reach out for help!