

# JAMBILIGHT



## Table of Contents

Debug info and log file: .....	2
Test setup:.....	2
The screen and regions: .....	3
Consolidated regions: .....	4
Margins: .....	5
Color weighing: .....	6
Color enhancement: .....	7
Color enhancement per channel:.....	7
Color correction (Intensity):.....	7
Resources: .....	8

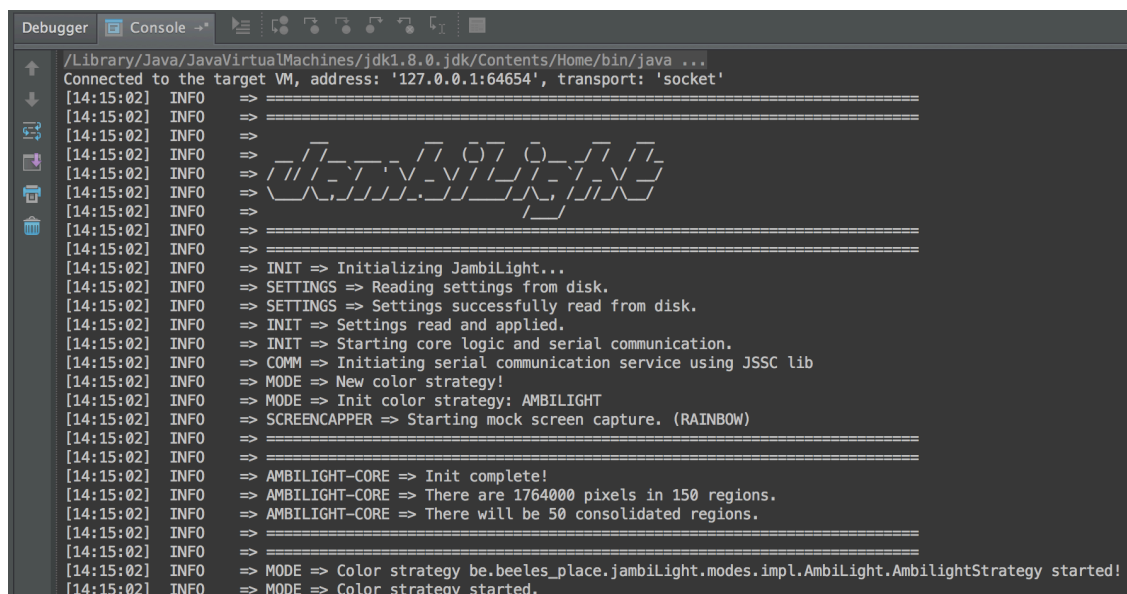
# JambiLight core information:

This document will briefly describe how the core mechanics of JambiLight work. This document only applies to the default implementation.

## Debug info and log file:

JambiLight supports extensive logging to track status and find issues.

Below is an example of the application startup log:



```
Debugger Console
/Library/Java/JavaVirtualMachines/jdk1.8.0.jdk/Contents/Home/bin/java ...
Connected to the target VM, address: '127.0.0.1:64654', transport: 'socket'
[14:15:02] INFO =>
[14:15:02] INFO =>
[14:15:02] INFO =>
[14:15:02] INFO =>
[14:15:02] INFO =>
[14:15:02] INFO =>
[14:15:02] INFO =>
[14:15:02] INFO =>
[14:15:02] INFO => INIT => Initializing JambiLight...
[14:15:02] INFO => SETTINGS => Reading settings from disk.
[14:15:02] INFO => SETTINGS => Settings successfully read from disk.
[14:15:02] INFO => INIT => Settings read and applied.
[14:15:02] INFO => INIT => Starting core logic and serial communication.
[14:15:02] INFO => COMM => Initiating serial communication service using JSSC lib
[14:15:02] INFO => MODE => New color strategy!
[14:15:02] INFO => MODE => Init color strategy: AMBILIGHT
[14:15:02] INFO => SCREENCAPPER => Starting mock screen capture. (RAINBOW)
[14:15:02] INFO =>
[14:15:02] INFO =>
[14:15:02] INFO => AMBILIGHT-CORE => Init complete!
[14:15:02] INFO => AMBILIGHT-CORE => There are 1764000 pixels in 150 regions.
[14:15:02] INFO => AMBILIGHT-CORE => There will be 50 consolidated regions.
[14:15:02] INFO =>
[14:15:02] INFO =>
[14:15:02] INFO => MODE => Color strategy be.beeles_place.jambilight.modes.impl.AmbiLight.AmbilightStrategy started!
[14:15:02] INFO => MODE => Color strategy started.
```

The application saves debug output to a file on disk. A new instance of this file is created **every time the application starts**. The name of the log file is as follows: **JambiLight[yyyy-MM-dd@HH-mm-ss].log**

In which the yyyy will be the year, the MM will be the month, the dd will be the day of the month, the HH will be the hours, the mm will be the minutes and the ss will be the seconds. **Per default only errors will be logged** in this file. This is to keep the size of the file minimal.

If you **start the application with the “enableDebug” parameter**, the application will **log all actions** to both the standard output (if running through a console or debug in IDE) and to the log file on disk. So instead of only logging the errors, it will also log all “info” and “debug” statements.

## Test setup:

In this document our “test setup” will look like this:

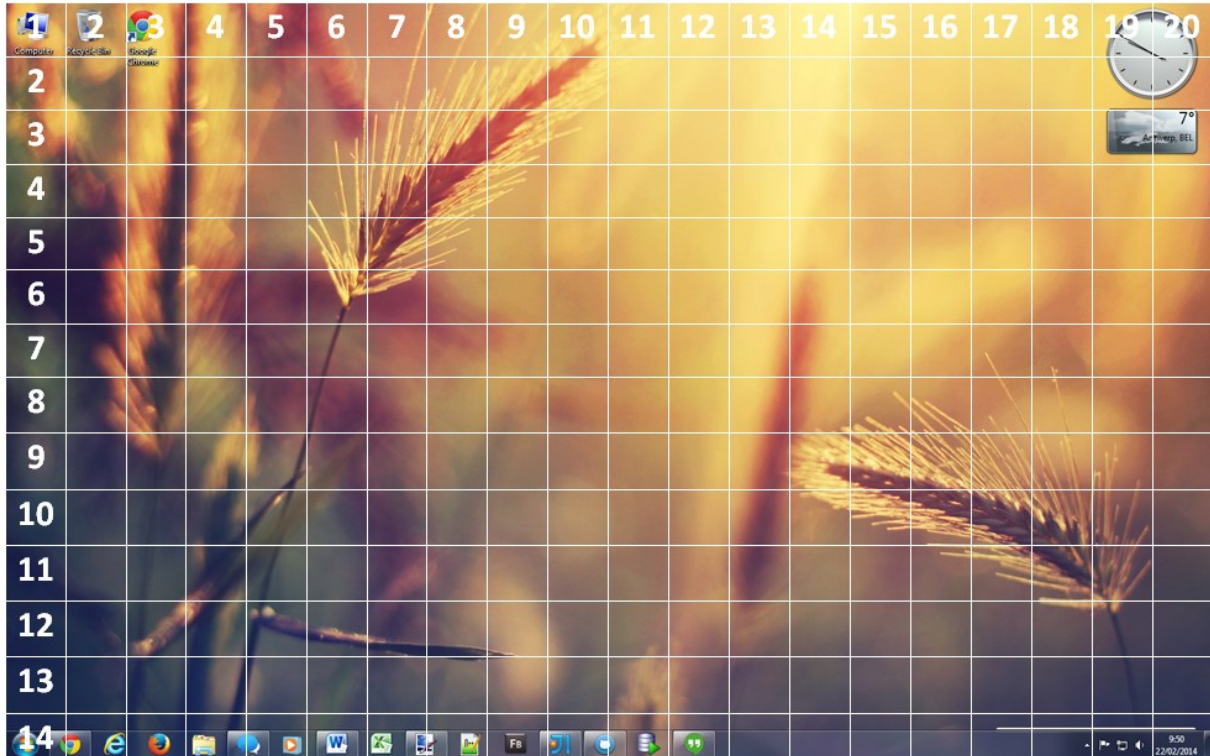
- Full HD screen (1920 horizontal and 1080 vertical pixels)
- 20 horizontal regions
- 14 vertical regions

### The screen and regions:

JambiLight divides the screen in regions. These regions are parameterized and can be user defined. The number of initial regions can be correlated directly to the amount of final consolidated regions later.

**When JambiLight starts it will divide the screen into virtual regions.** The specified horizontal and vertical regions calculate the total amount of regions.

And this is how the division of the screen would look like.



## Consolidated regions:

As the goal of JambiLight is to drive RGB LEDs we don't need the 280 regions that will be calculated and averaged by the JambiLight core. As LEDs will only be placed at the edges of the screen this is where the consolidated regions come in to play.

**Right now each consolidated region is mapped directly to a RGB LED.** In future versions support may be added to span one region over multiple LEDs.

**The amount of consolidated regions can be calculated as follows:**

***Consolidated regions = horizontal regions + (vertical regions – 2) + horizontal regions + (vertical regions – 2)***

In this example that would give:  $Cr = 20 + (14 - 2) + 20 + (14 - 2) = 64$

The reason that the vertical regions have to be subtracted by 2 each time is simple. The horizontal "bars" lie below and sit on top of the vertical regions. For this reason (and the UI) please use a **minimum size for both regions of 4**. There is **no limit to the maximum amount of horizontal and vertical regions** (in theory, in the real world numbers too high will make the application take too long and become unresponsive). In the UI you enter either the total number of LEDs (which is equal to the amount of consolidated regions) or you enter the horizontal and vertical amount of LEDs. (So the formula would become: horizontal LEDs \* 2 + vertical LEDs \* 2)

When the regions are consolidated the unused regions in the center of the screen are not thrown away. They are used to get a better average color per region and thus they are consolidated into one region.

Below is an image that will be explained on the next page. It shows how the consolidation algorithm works.





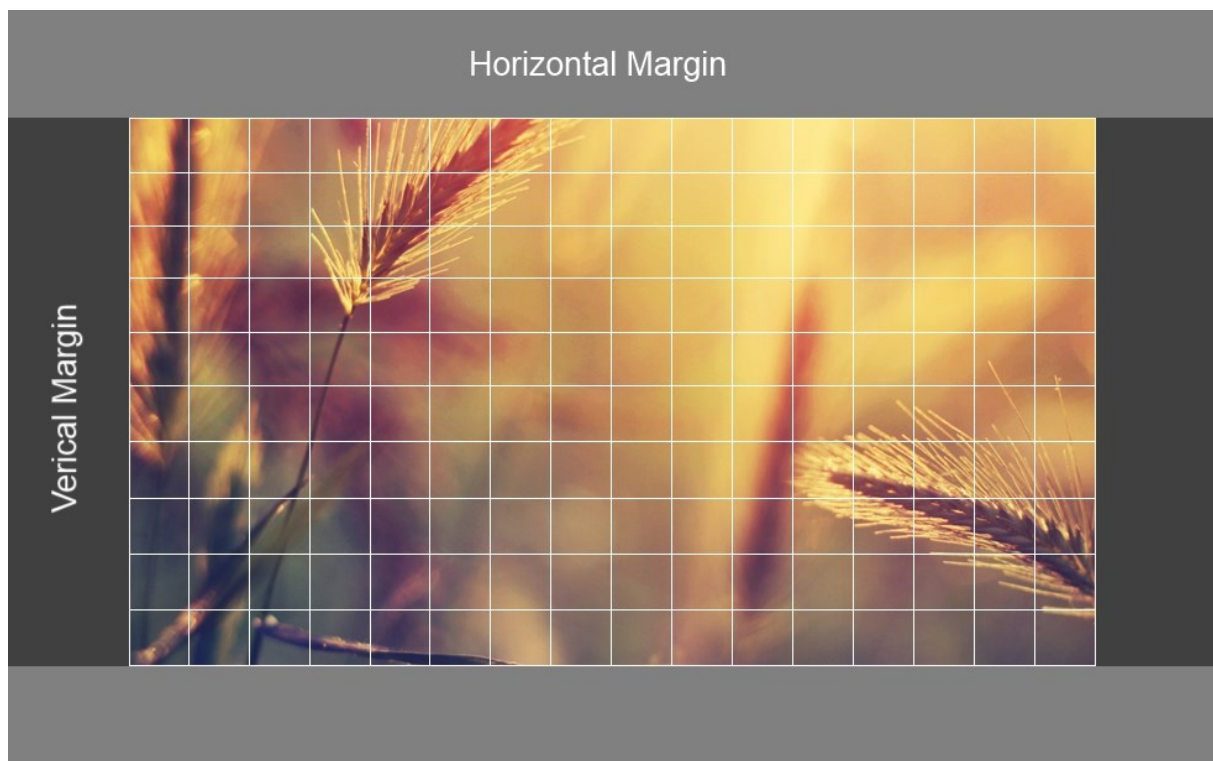
The consolidation of the regions takes place in several steps:

- A. All columns in Q1 and Q2 are consolidated into one region per column.
- B. All columns in Q3 and Q4 are consolidated into one region per column.
- C. All rows in Q1 and Q3 are consolidated into one region per row.
- D. All rows in Q1 and Q3 are consolidated into one region per row.
- E. The 4 corners in Q1, Q2, Q3 and Q4 are recalculated by their nearest already consolidated neighbors.

The result is that 64 consolidated regions now remain and each consolidated region takes several “normal” regions into account for its final color. The old regions are discarded and the new consolidated regions are returned for further use in the JambiLight core.

### Margins:

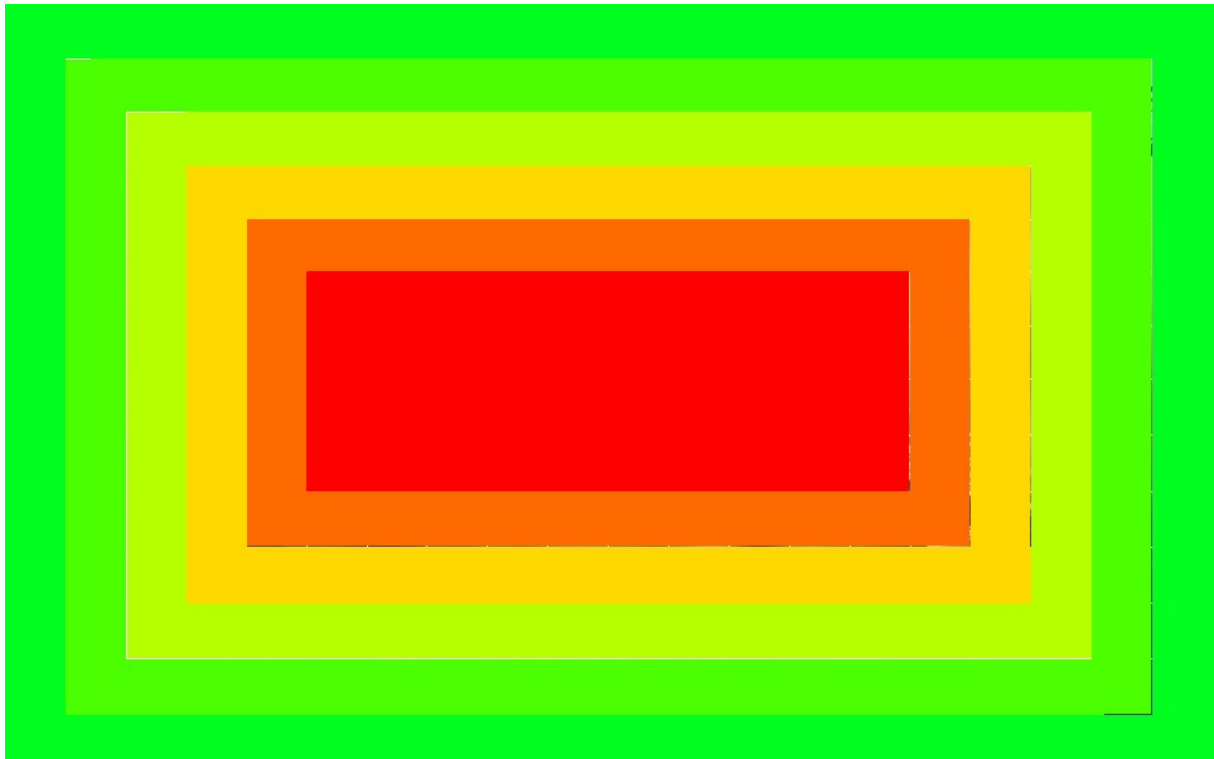
The JambiLight application knows the concept of margins. Both **horizontal and vertical margins can be specified**. Margins can be handy when certain areas of the screen are a constant unwanted color (say the black bars with movies).



Margins are taken into account during the region consolidation process. **Steps A and B take the horizontal margin into account, while steps C and D take the vertical margin into account.** Using margins does not change the final amount of consolidated regions.

### Color weighing:

JambiLight has a parameter which when enabled will make it so that **regions can have a weight assigned to them** according to their importance on the screen. And **those weights are then used during the consolidation of the regions**. This will make it so that the colors on the edges of the screen are more important than those in the center of the screen. Color weighing can be completely disabled if required.



**The image above shows this order.** The region closest to the edge has the most importance. The importance goes down with one or more levels depending the chosen weight step size (The weight goes down until the minimum threshold is reached, then it stays the same => red square). **The image shown here is pure informational**, as it does not represent the weights during the consolidation (The image would change for the individual steps during the consolidation).

### Color enhancement:

The color enhancement feature of JambiLight allows for colors to be enhanced. It will detect the base color type (red/green/blue/yellow/cyan/purple/white/grey/black) and apply a multiplication to that color base type to make it more pronounced.

**Color enhancement is applied before the regions are consolidated.** This allows the more pronounced color to be consolidated better later on.

This feature has **one parameter** that defines **how much the detected color should be amplified.**

### Color enhancement per channel:

This sub feature of the color enhancement feature **can be enabled if color enhancement is also enabled.**

This feature **will run after the initial color enhancement has been performed** and will **multiply the individual red, green and blue color channels with the values that are given.** This can be used to strengthen certain colors or suppress them.

This feature has **three parameters, a multiplication value per color channel.**

### Color correction (Intensity):

Color correction is an intensity correction that can be enabled to correct the intensity of the colors for each consolidated region (thus this **happens after region consolidation**).

This feature was implemented because when a certain color is displayed, say red, it will be less bright than white. **This is because to display a red color only one LED is used while when displaying white all three R/G/B LEDs are used to display that color.** This creates an inequality in light intensity making event grey (dimmed white light) too bright when compared to the other colors.

**This feature has three parameters** that can be edited to accommodate different user preferences. In short the **scale up value for colors** can be changed, the **scale down value for white/grey** can be changed and the **detection threshold for white/grey** can also be changed.



## Resources:

The following resources were used:

Front cover image → <http://www.flickr.com/photos/solarbotics/7832369412/sizes/l/>