

Thomas Edison MV

*A customizable lighting system for RPG Maker MV. Created by **Soulpour777**.*

Website: soulxregalia.wordpress.com

Current Version: 1.0. Date Finished: 9 / 10 / 2016



Table of Contents:

Introduction

Installation

Creating a Classical Light

Creating a Custom Light

Explaining the Parameters

Examples and Light Shows

Support and Feedback

Introduction

Thomas Edison MV is a plugin that allows the developer to create beautifully crafted lights for their game. The idea behind this plugin was to revive the popular VX script of the same name by BulletTXT. This plugin however has been modeled and developed with the combination of ES5 and ES6 with a few differences with the old VX counterpart.

So, what does this plugin offer? Just creating lights?

Yes, that's basically it. That way, it allows you to make your maps more beautiful. The idea of this plugin is taken from playing around the tones and blending of an object. Shadows and such are not found and created by this plugin.

Can we use different image for every light?

Yes. I made the plugin like that so every light and sprite tied to it can make different types and shapes of the light depending on how they are displayed on the map. Why was it designed so? The reason behind this aside from this is if you're creating a parallax map, normally lighting systems follow the tileset (if programmed so) instead of the current displayed images. That works for this purpose. If you're using a parallax image, it'd be best that the light would also follow the image to create a more realistic effect.

Why are the lights tied to a switch?

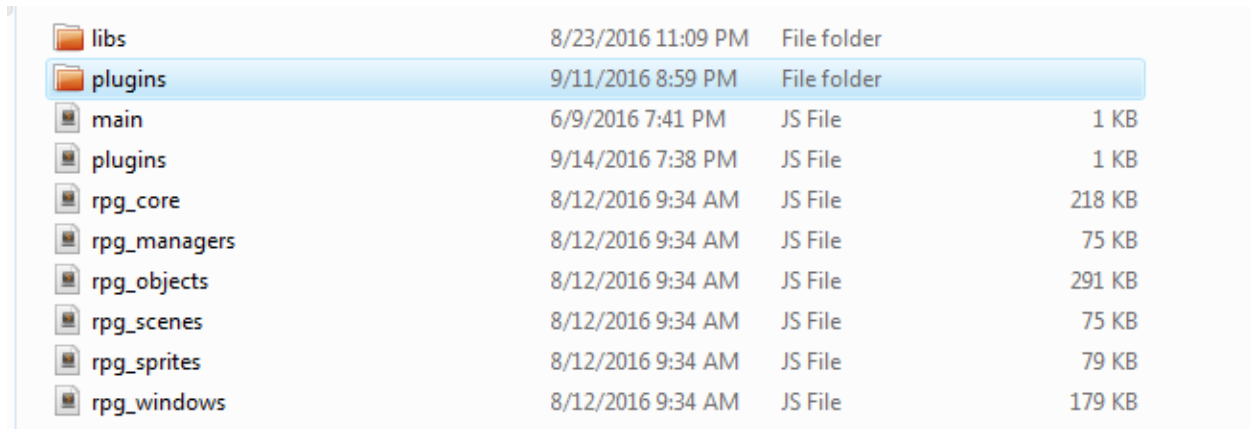
The lights are tied to a switch so you can create stop lights or lights that would only occur during a said timing. With this, you can create traffic / signal lights as well as use it to create some kind of light puzzle event. Not only will this plugin serve as your lighting system but also for puzzles. Cool, right?

If so, isn't this doable in events?

Unfortunately, you can't, not unless you tie everything via script commands, then yes. But for easier take and quick build, making lights using a plugin where everything is set properly is better. I believe that with this plugin, it would allow you to quickly make something as making lights faster than eventing.

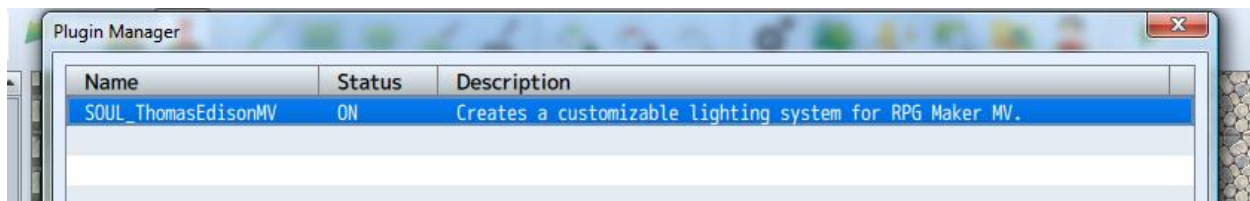
Installation

Place the plugin on your js / plugins folder.



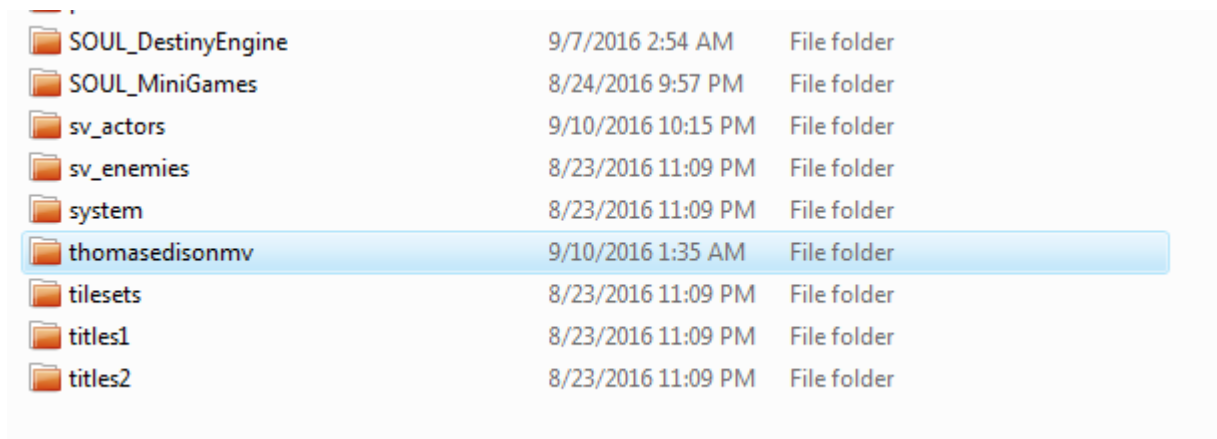
libs	8/23/2016 11:09 PM	File folder	
plugins	9/11/2016 8:59 PM	File folder	
main	6/9/2016 7:41 PM	JS File	1 KB
plugins	9/14/2016 7:38 PM	JS File	1 KB
rpg_core	8/12/2016 9:34 AM	JS File	218 KB
rpg_managers	8/12/2016 9:34 AM	JS File	75 KB
rpg_objects	8/12/2016 9:34 AM	JS File	291 KB
rpg_scenes	8/12/2016 9:34 AM	JS File	75 KB
rpg_sprites	8/12/2016 9:34 AM	JS File	79 KB
rpg_windows	8/12/2016 9:34 AM	JS File	179 KB

After placing it inside the folder, remember to install the plugin on your Plugin Manager. Don't forget to turn it ON to work.



Please do not rename the plugin.

Next, place the light images on the img / thomasedisonmv folder. If it is not existent, create one.



SOUL_DestinyEngine	9/7/2016 2:54 AM	File folder	
SOUL_MiniGames	8/24/2016 9:57 PM	File folder	
sv_actors	9/10/2016 10:15 PM	File folder	
sv_enemies	8/23/2016 11:09 PM	File folder	
system	8/23/2016 11:09 PM	File folder	
thomasedisonmv	9/10/2016 1:35 AM	File folder	
tilesets	8/23/2016 11:09 PM	File folder	
titles1	8/23/2016 11:09 PM	File folder	
titles2	8/23/2016 11:09 PM	File folder	

You can create many light images as many as you want. You can use them individually later on.

Creating a Classical Light

What is a Classical Light?

A classical light is a setting of the light images that you have limited controls of. This acts also as a guide for the plugin user on how the lights behave.

To create one, place this in an event comment:

LIGHT switch_id tone flicker scale

Where:

switch_id is the id of the switch the classical light is tied to in order to work. This means that the light image would only be displayed if the switch is turned on.

tone is the image's tone color. This goes in the form of an array, something like this: [122,255,123,100]. The numbers represents the needed factors in the tone: [r,g,b,gray]. Please make sure that you don't have any spaces when writing them. The plugin is space sensitive to tones.

flicker

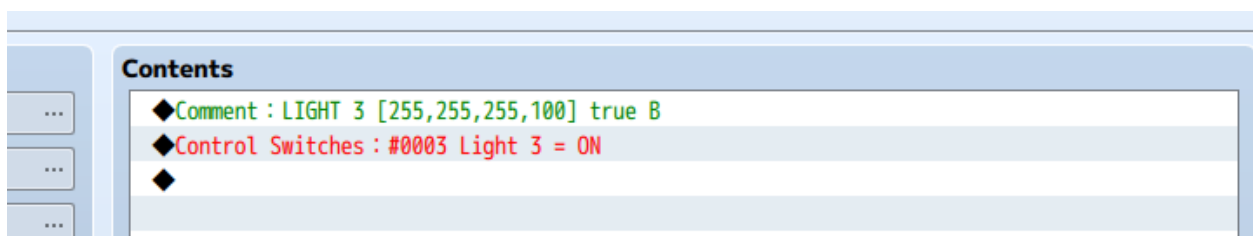
Flicker is set either true or false as values. When set to true, the light would flicker and would have changes to the visibility and on its behavior.

scale

There is a difference between the scale values when you use the classical LIGHT or the CUSTOM light. When you're using the classical light, the values should either be of the following:

- A - normal scale. Not zoomed.
- B - zoomed twice. The light is bigger.
- C - zoomed four times.
- D - zoomed six times.
- E - zoomed eight times.

Here is an example how the light is made:



This will show the light image, flickering and scaled twice when switch 3 is on. The event is set to parallel, so after turning switch 3, it works.

It looks like this:



You can play around the tones and you can see different types of them as you change the color (from positive to negative values).

Creating a Custom Light

Creating a Custom Light is pretty much similar to how a classical light is made, however with new functions you can play around.

Place this in a comment in an event command:

CUSTOM switch_id image ax ay tone flicker scale blend opacity

Where:

ax / ay

ax and ay represents the x and y value dimensions of the light image. Reason for this is because when the image is scaled twice or thrice, the image actually needs adjustment where it is shown. Ax and Ay behaves like a margin. You can use it to indent the images to their original position.

IMAGE

If you're using the CUSTOM light, this parameter is needed. this is the name of the image you're going to use as the light source.

BLEND

If you're using the custom light, this parameter is needed. The blend parameter can only be any of the following:

NORMAL - sets the blend mode to normal.

ADD - sets the blend mode to Add.

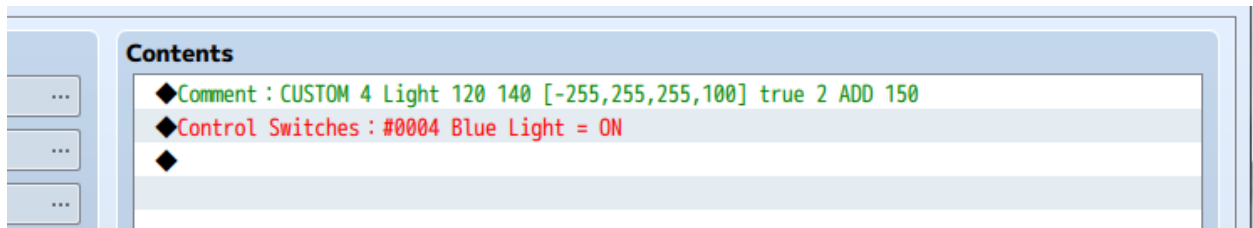
SUBTRACT - sets the blend mode to Subtract.

SCREEN - sets the blend mode to Screen.

OPACITY

if you're using the custom light, this parameter is needed. The opacity parameter holds how visible your light is in the screen. The opacity parameter should be in numbers.

Here is an example of a custom light:



This custom light would only appear if switch 4 is turned on. This reflects a Bluish light flickering and scaled twice.

It looks like this:



You can play around the different tones to have different results.

Explaining the Parameters

Let me explain a few things about the parameters. The first section is the Default Scales. You see, when you do scaling, this means that you're going to zoom the image. When the scaling is 1, this is the exact size of the image. When you scale the image to 2, then the image gets doubled. It follows this for every arrangement you make.

Parameters	
Name	Value
-- DEFAULT SCALES--	
Scale A	1
Scale B	2
Scale C	4
Scale D	6
Scale E	8

The scale on the event comment for the classical light is grouped to ABCD and E. These are the settings of what scaling they are prepared.

There are two extra parameters you will need to consider for the Classical Light. This would be its opacity and flicker rate.

-- CLASSICAL LIGHT--	
Classical Light Opacity	130
Classical Light Flicker Rate	20

The flicker rate is how the light would behave or flicker in the map to simulate realistic light whenever it is activated.

Last is the scale adjustment.

-- SCALE ADJUSTMENT--	
Scale A XCoord	60
Scale A YCoord	100
Scale B XCoord	120
Scale B YCoord	140
Scale C XCoord	250
Scale C YCoord	200

The scale adjustment for the Classical Light is the margin / indentation of the image whenever they are displayed on the map. Originally, each light is tied to the direction of where the event it was called from, but when the image is scaled, it actually changes the proper direction where you can place it. Therefore, these parameters would allow you to adjust them depending also on the image that you're using.

Examples and Light Shows

Here are a few examples on Classical Lights:



The code for this is: LIGHT 2 [255,255,255,100] true B

This would depend on which switch you are tying it on.



The code for this is: LIGHT 6 [255,-155,-155,100] true C

Here is a Green type of light for Custom Lights:



The code for this is:

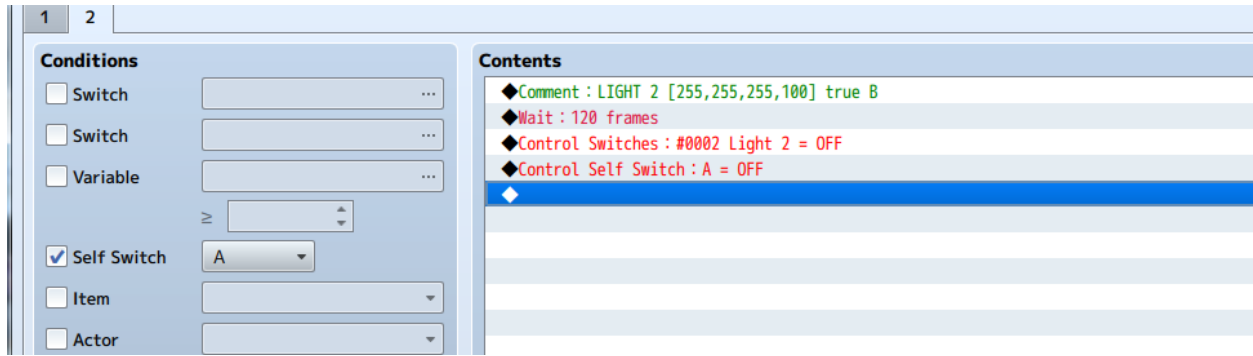
CUSTOM 5 Light 120 140 [-155,255,-155,100] true 2 ADD 150

Please refer to the creating process what the numbers mean.

Now, let me create a stop light / traffic light or you can say a light show. First, create the type of light. Make sure that the event is turned to parallel process. The event commands are something like this:

Conditions		Contents
<input type="checkbox"/> Switch	...	◆Comment : LIGHT 2 [255,255,255,100] true B
<input type="checkbox"/> Switch	...	◆Wait : 60 frames
<input type="checkbox"/> Variable	...	◆Control Switches : #0002 Light 2 = ON
	≡	◆Control Self Switch : A = ON
<input type="checkbox"/> Self Switch	▼	◆
<input type="checkbox"/> Item	▼	

Now, create a new tab of the event. Make sure that the condition is turned when Self Switch A is on. This event here would allow us to show the light, wait for 60 frames and goes to self switch A based tab. What happens there is something like this:



Make sure that the event is turned parallel process as well. What happens with this light is that after it waits for 60 frames, you turn the light off and then waits for 120 frames, turns the light on again. This way, you can create traffic lights!

Support and Feedback

If you like this plugin, please don't forget to give the video (youtube video) a thumbs up and if you really like my works, please drop a pledge on Patreon, it helps a lot. This way, we can build more awesome plugins!

Patreon: <https://www.patreon.com/Soulpour777>

For any errors, feedback and anything of the sort, please don't hesitate to message me on my blog: <https://soulxregalia.wordpress.com/>

Thank you very much!