# Magic Light Show Developer's Note

"Magic Light Show" is not so much a stage for a light show as it is a development tool for light shows. Its core is a lighting control program written in VAM-Scripter based on the "__MainScript" atom. This program allows users to achieve flexible control of one or multiple lighting groups, or even individual lights within a lighting group, by writing "configuration text" and parsing and executing it, thus achieving complex and variable lighting effects. This greatly facilitates the design and choreography of light show stages.

**Basic concepts:**

1. Lighting: The lighting used in the light show performance is not a real light source, but rather volume light developed by PluginIdea. Using real light sources would be too taxing on the system, as hundreds of lights are needed.
2. Lighting group: Typically refers to a group of highly coordinated lights that belong to the same "group control parent atom" ("_Dome/CX_CC", where X is a sequential number starting from 0). All the lights in a lighting group share a "volume light" plugin, with consistent properties such as light cone angle, brightness, and color.
3. Individual light: Each specific light within a lighting group. Its atomic name is "_Dome/CX_LY" (where X and Y are sequential numbers starting from 0), and its parent atom is "_Dome/CX_CC". In the naming convention for lights, "CX" indicates the lighting group it belongs to, and "LY" indicates the specific light. The tilt (Pitch) rotation, azimuth (Yaw) rotation, and on/off status of the light can be controlled independently, while other attributes

follow the lighting group.

## Description of attributes and parameters:

| Objects | Attributes | Parameters | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | delay | duration | speed | mode | WU | sVal | eVal | CD | vuml |
| **Light** *each single light* | **Pitch** | Float | Float | Float | Int 0,1,2,3 | Float | Float 0~180 | Float 0~180 | Float | String |
| | **Yaw** | Float | Float | Float | Int 0,1,2,3 | Float | Float -180~180 | Float -180~180 | Float | String |
| | **YawAlt** | Float | Float | Float | Int 0,1,2,3 | Float | Float -90~90 | Float -90~90 | Float | String |
| | **Switch** | Float | Ignore | Ignore | Int 0,1,2,3 | Float | Int 0,1 | Int 0,1 | Float | String |
| **Circle** each group of lights | **Intensity** | Float | Float | Float | Int 0,1,2,3 | Float | Float 0~3 | Float 0~3 | Float | String |
| | **Angle** | Float | Float | Float | Int 0,1,2,3 | Float | Float 0~50 | Float 0~50 | Float | String |
| | **Radius** | Float | Float | Float | Int 0,1,2,3 | Float | Float 0~0.1 | Float 0~0.1 | Float | String |
| | **FadeEnd** | Float | Float | Float | Int 0,1,2,3 | Float | Float 0~60 | Float 0~60 | Float | String |
| | **Color** (RGB) | Float | Float | Float | Int 0,1,2,3 | Float | Float X 3 0~255 | Float X 3 0~255 | Float | String |
| | **Color** (HSL) | Float | Float | Float | Int 0,1,2,3 | Float | Float X 3 0~360 0~100 0~100 | Float X 3 0~360 0~100 0~100 | Float | String |
| | **CParm1 CParm2 CParm3** (RGB) | Float | Float | Float | Int 0,1,2,3 | Float | Float 0~255 | Float 0~255 | Float | String |
| | **CParm1** (HSL) | Float | Float | Float | Int 0,1,2,3 | Float | Float 0~360 | Float 0~360 | Float | String |
| | **CParm2 CParm3** (HSL) | Float | Float | Float | Int 0,1,2,3 | Float | Float 0~100 | Float 0~100 | Float | String |

## Attributes:

1. **Pitch**: The tilt angle of the light, with a rotation range of 0~180 degrees;
2. **Yaw**: The horizontal rotation angle of the light, with a rotation range of -180~180 degrees;
3. **YawAlt**: The custom plane rotation angle of the light, with a rotation range of -90~90 degrees. This means that after determining a plane through the **Pitch** and **Yaw** attributes (Note 1: the plane determined by the light direction and the rotation axis of **Pitch**), the light is rotated in this plane. It should be noted that the YawAlt attribute only takes effect after the Pitch and Yaw movements have ended.
4. **Switch**: The on/off status of the light;
5. **Intensity**: Brightness, with a value range of 0~3;
6. **Angle**: Light cone angle, with a value range of 0~50;
7. **Radius**: Size of the light ball entity, with a value range of 0~0.1;
8. **FadeEnd**: Light cone fade-out distance, with a value range of 0~60;
9. **Color**: Light color, defaulting to RGB, with R (red), G (green), and B (blue) value ranges of 0~255; it can also support HSL color, by adding a "type" parameter under the **Color** attribute and setting its value to "HSL". The value ranges of the variables are H (hue) 0~360, S (saturation) 0~100, L (lightness) 0~100;
10. **Color separation mode**: As the name suggests, color separation mode is to split the color of a group of lights into three independent attributes and control them separately. For example, in RGB mode, the color is split into three independent attributes: R (red), G (green), and B (blue), corresponding to CParm1, CParm2, and CParm3; in HSL mode, the color is split into three independent attributes: H (hue), S (saturation), and L (lightness), also corresponding to CParm1, CParm2, and CParm3. Each independent attribute can be set with

different change speeds, loop modes, wait times, warm-up and cool-down times, etc., thus achieving more colorful color changes through combinations of the three independent attributes. The setting method is to describe the CParm1, CParm2, and CParm3 attributes separately in the configuration text instead of describing the Color attribute.

Note 1: The rotation result of **Yaw** determines the rotation axis of **Pitch**, and the rotation result of **Pitch** determines the light direction perpendicular to the rotation axis. These two lines determine a plane. The role of the **YawAlt** attribute is to rotate the model light on the aforementioned plane without always maintaining perpendicularity to the rotation axis of **Pitch**.

**Parameters:**

1. **delay**: Delay time, which is how many seconds to wait before starting to process;
2. **duration**: Duration, which is how much time it takes for the relevant attribute value to gradually change from the start value (sVal) to the end value (eVal). This parameter is invalid for the Switch attribute;
3. **speed**: Change speed, which is how much the relevant attribute changes per second. This parameter is only effective when "duration" is "-1" and is invalid for the Switch attribute;
4. **mode**: Action mode. "0" for single execution (A2B); "1" for repeated execution (A2B, A2B, A2B...); "2" for single forward and reverse (A2B, B2A); "3" for repeated forward and reverse (A2B, B2A, A2B...);
5. **WU**: Warm-up time, which is how many seconds to wait after the relevant attribute is assigned the start value (sVal) before starting to change. Warm-up time is part of the action and will be included in various action modes;
6. **sVal**: Start value. Users can specify the start value for relevant attributes or use the current

value of the attribute by setting sVal to "999" (this setting also applies to the Color attribute, i.e., set to "999,999,999");

7. **eVal**: End value. For the Switch attribute, if eVal is a negative integer, it means that the on/off status of the light is obtained randomly according to a certain probability. If eVal is "-1", the probability of the light being on is 10%, and if eVal is "-2", the probability is 20%, and so on;
8. **CD**: Cooldown time, and after the related attribute reaches the termination value (eVal), wait for how many seconds before the action ends. Cooldown time is part of the action and will be included in various action modes.
9. **vuml**: External vuml variable, which comes from the "plugin#0_JayJayWon.VUML" plugin on the "__MainScript" atom, is a floating-point variable (such as "vFLOAT1Result", and the variable value should be between 0 and 1) or a Boolean variable (such as "vBOOL1Result". Boolean variables are only applicable to specify Switch properties as on or off). When a certain attribute specifies a vuml parameter, then the value of that attribute will follow the specified floating-point variable to change between sVal and eVal. This status is a continuous state unless there is new instruction input and execution. vuml parameters are also applicable to Color attributes.

**Universal Random Parameter Settings**

In addition to the conventional operations on parameters introduced above, it is also possible to perform universal random settings for the sVal and eVal parameters (except for Switch attributes). The specific method is to set sVal or eVal to "-9XY". X indicates the starting point of the parameter value domain (from the smallest to the largest) based on the fraction (10% as a block), and Y indicates the size of the random domain, which is equivalent to a certain percentage of the parameter value domain (also 10% as a block, with 0 indicating 100%). When entering the action for the first time, the system will randomly generate a value within the specified range and assign it to sVal or eVal.

**Universal Incremental Settings**

To facilitate users in setting a parameter (such as delay) of a certain attribute of a group of lights or a certain light to increase or decrease sequentially (of course, the order can be specified according to the user's needs), "Magic Light Show" provides a universal incremental function. This function supports most parameters of all attributes, including delay, duration, speed, WU, sVal (not applicable to Switch and Color), eVal (not applicable to Switch and Color), CD, and vuml.

The specific setting is like "delay":"0#0.1", where "#" is the starting quantity before the parameter, and "#" is the amount of incremental increase after each time. The increment can be negative. This writing method is applicable to all floating-point parameters. For character-type vuml parameters, a different writing method is required. Specifically, it is written as "vuml":"vFLOAT#Result#16". The incremental setting results obtained are in order: "vFLOAT16Result", "vFLOAT17Result", "vFLOAT18Result", "vFLOAT19Result", "vFLOAT20Result" ...

**Configuration Text**

The format of configuration text follows that of JSON, which is relatively concise and easy to understand. However, due to the limitations of the development environment, it is difficult to further optimize. Specifically, it looks like:

```
Circle|All
Light|All
=
Pitch|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":0, "eVal":0, "WU":0, "CD":0, "vuml":""
Yaw|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":999, "eVal":0, "WU":0, "CD":0, "vuml":""
YawAlt|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":999, "eVal":0, "WU":0, "CD":0, "vuml":""
Switch|"delay":"0#0.1", "duration":0, "speed":0, "mode":1, "sVal":1, "eVal":0, "WU":"24#-0.2", "CD":"0#0.2",
"vuml":""
=
Angle|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":999, "eVal":1, "WU":0, "CD":0, "vuml":""
Intensity|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":999, "eVal":1, "WU":0, "CD":0, "vuml":""
Radius|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":999, "eVal":0.05, "WU":0, "CD":0, "vuml":""
FadeEnd|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":0, "eVal":10, "WU":0, "CD":0, "vuml":""
Color|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":"-900,-900,-900", "eVal":"-900,-900,-900", "WU":0,
"CD":0, "vuml":""
```

In the above example, a series of delay settings were applied to the Switch attribute, and random settings were applied to the Color attribute.

For another instance：

```
Circle|0
Light|3,9
Circle|1
Light|4,5,13,14
Circle|2
Light|5,6,7,17,18,19
```

```
Circle|3
Light|6,7,8,9,21,22,23,24
Circle|4
Light|7,8,9,10,11,25,26,27,28,29
=
Pitch|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":0, "eVal":90, "WU":0, "CD":0, "vuml":""
Yaw|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":0, "eVal":0, "WU":0, "CD":0, "vuml":""
YawAlt|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":0, "eVal":0, "WU":0, "CD":0, "vuml":""
Switch|"delay":0, "duration":0, "speed":0, "mode":1, "sVal":1, "eVal":0, "WU":3, "CD":3, "vuml":""
=
Angle|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":20, "eVal":30, "WU":0, "CD":0,
"vuml":"vFLOAT1Result"
Radius|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":0.01, "eVal":0.05, "WU":0, "CD":0, "vuml":""
Intensity|"delay":0, "duration":1.5, "speed":0, "mode":3, "sVal":0, "eVal":1, "WU":0, "CD":0, "vuml":""
FadeEnd|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":0, "eVal":10, "WU":0, "CD":0, "vuml":""
Color|"delay":0, "duration":0, "speed":0, "mode":0, "type":"HSL", "sVal":"0,0,100", "eVal":"360,100,100", "WU":0,
"CD":0, "vuml":""
```

The above example demonstrates how to configure different lights in different groups simultaneously. It can be seen that the configuration flexibility is very high, which gives the user a broad creative space, but also means a relatively higher understanding difficulty and learning cost. In addition, to achieve the desired effect, users need to constantly try and optimize in the creative process. Therefore, it takes a certain threshold to use "Magic Light Show" deeply.

Here is a commonly used initialization configuration for easy copy & paste.

```
Circle|All
Light|All
=
Pitch|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":0, "eVal":0, "WU":0, "CD":0, "vuml":""

Yaw|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":0, "eVal":0, "WU":0, "CD":0, "vuml":""

YawAlt|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":0, "eVal":0, "WU":0, "CD":0, "vuml":""

Switch|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":0, "eVal":0, "WU":0, "CD":0, "vuml":""

=

Angle|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":999, "eVal":10, "WU":0, "CD":0, "vuml":""

Intensity|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":999, "eVal":1, "WU":0, "CD":0, "vuml":""

Radius|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":999, "eVal":0.05, "WU":0, "CD":0, "vuml":""

FadeEnd|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":0, "eVal":10, "WU":0, "CD":0, "vuml":""

Color|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":"0,0,0", "eVal":"255,255,255", "WU":0, "CD":0,
"vuml":""
```

**Configuration and Execution**

The basic working principle of "Magic Light Show" is to write configuration text, then parse the configuration data and update object parameters. Then, every frame (physical frame), the system will operate and maintain the corresponding objects based on the changes in parameters.

Users can operate the "plugin#1_Scripter" plugin of "__MainScript" atom to write configuration text through any external trigger instruction to the "Config String" variable. Then, users can execute the "Execute New Action" method of the "plugin#1_Scripter" plugin of "__MainScript" atom to make the configuration text be parsed and executed at the end of the physical frame.

"Writing configuration" and "execution" are a pair of actions that need to be completed continuously. Users can continuously perform multiple "write configuration" and "execute" operations in the same frame to achieve complex operations for different lights and different attributes.

Configuration is allowed to overwrite. If there are multiple configurations for the same object and attribute in the same frame, only the last configuration will be effective. Therefore, a small trick can be introduced:

After a light completes an action, in order to ensure a clean start for the next action, we will execute a "ResetAll". However, if we immediately execute the next action in the same frame, there may be cases where the state cannot be refreshed cleanly. The reason is that in the same frame, one light receives the configuration information of RestAll after waiting for the execution refresh (really ending). If at this time, subsequent actions also transmit configuration information to this light, it will overwrite the configuration information transmitted by RestAll. In most cases, immediately executing subsequent actions will not cause problems. However, if the subsequent action requires the light to delay first, then the state of the previous action of the light will be retained. It feels like the state did not refresh cleanly.

Therefore, for insurance purposes, after executing RestAll, it is recommended to wait for a short time before executing the subsequent action. At least leave one frame's time for RestAll to refresh the light state cleanly.

**Detailed Action Execution Process**

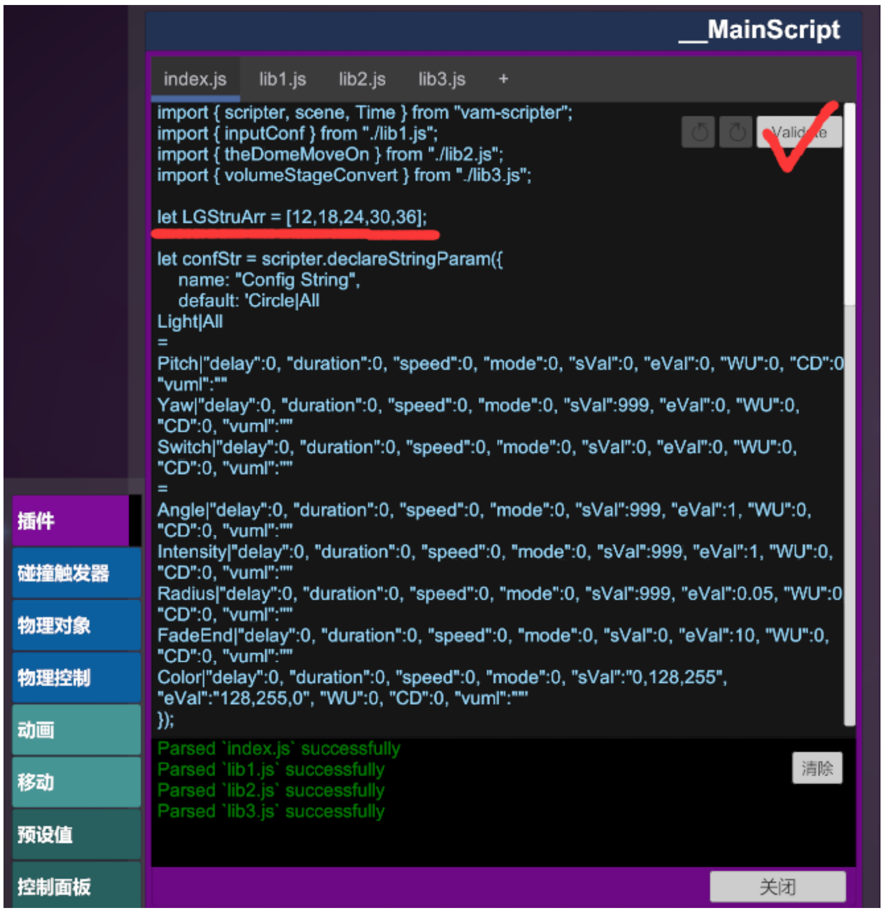
The execution of each action follows this process:

Start → Wait (delay) → Set initial value (sVal) → Warm-up (WU) → Start value change → Achieve termination value (eVal) → Cool down (DC) → End

In any action mode, repeated operations and reverse operations of the action do not include the "Wait (delay)" part. Reverse operation refers to the reverse of all processes except for the "Wait (delay)" part, not just the "value change" part. Understanding this action process will be helpful for rational configuring and programming to achieve desired effects.

**Advanced Extension**

If you want to create your own light group or brand-new stage, in addition to creating the group and lights according to the naming rules, you also need to adjust some VAM-Scripter code. Of course, this is also very simple, just adjust one line of code.



```
import { scripter, scene, Time } from "vam-scripter";
import { inputConf } from "./lib1.js";
import { theDomeMoveOn } from "./lib2.js";
import { volumeStageConvert } from "./lib3.js";

let LGStruArr = [12,18,24,30,36];

let confStr = scripter.declareStringParam({
    name: "Config String",
    default: 'Circle|All
Light|All
=
Pitch|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":0, "eVal":0, "WU":0, "CD":0
"vuml":""
Yaw|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":999, "eVal":0, "WU":0,
"CD":0, "vuml":""
Switch|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":0, "eVal":0, "WU":0,
"CD":0, "vuml":""
=
Angle|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":999, "eVal":1, "WU":0,
"CD":0, "vuml":""
Intensity|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":999, "eVal":1, "WU":0,
"CD":0, "vuml":""
Radius|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":999, "eVal":0.05, "WU":0
"CD":0, "vuml":""
FadeEnd|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":0, "eVal":10, "WU":0,
"CD":0, "vuml":""
Color|"delay":0, "duration":0, "speed":0, "mode":0, "sVal":"0,128,255",
"eVal":"128,255,0", "WU":0, "CD":0, "vuml":""'
});
```

```
Parsed `index.js` successfully
Parsed `lib1.js` successfully
Parsed `lib2.js` successfully
Parsed `lib3.js` successfully
```

Each number represents the number of lights in the lamp group. I think you understand:-) Don't forget to click the checked button after changing it.

Of course, there is still a lot of knowledge about lamp group creation itself, which will not be expanded here. We can discuss it later if we have the opportunity.