

Smart Material Changer

'Amber

版本: 1.12

[PDF 文档](#)

简介

- 基于脚本配置而不是notecard。更快的加载/传输速度，更自由的书写。
- 内核与产品功能是分离。可以支持菜单、HUD形式，本地与远端控制。
- 容易扩展且没有硬性束缚。
- 智能的匹配规则。

Ps: 没有使用 Notecard 作为配置的载体，是因为丫加载实在是太慢了，太他妈的慢了，实在是太他妈的慢了。

脚本列表

发送端（内核）

| 脚本 | 说明 |
|------------|--------------|
| SMC.KERNEL | 内核，材质管理器，存储器 |
| .SMC | 用于KERNEL的配置 |

客户端（加载器）


| 脚本 | 说明 |
|-------------|------------------------------------|
| SMC.Client | 材质配色应用器，放在需要被替换材质的物体，接受KERNEL发出的信息 |
| .SMC.Client | 用于SMC.Client的配置 |

其他

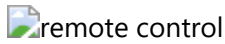
| 脚本 | 说明 |
|-----------------|--|
| SMC.HUD.TRIGGER | HUD专用，将Linkset中，Prim的描述以 PART.SET 的格式来发起材质替换 |
| SMC.Menu | 通过点击弹出菜单，选择PART与SET，实现材质的替换 |
| .SMC.Menu | 用于SMC.Menu的配置 |

示意图

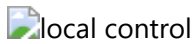
脚本关系

script relationship

远端



本地



远端/本地

多重部署分区控制

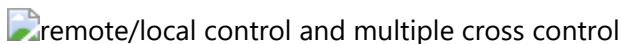
- 同一个 linkset 中的不同 prim 中可以分别放置多个 SMC.Client，他们可以负责各自的部位，被一个或者多个 SMC.KERNEL 控制
- 同一个 linkset 中的不同 prim 中可以分别放置多组 SMC.KERNEL + SMC.Client，通过本地的方式分别控制多组规则



远端/本地 交叉

多重部署分区交叉控制

- SMC.KERNEL 与 SMC.Client 之间使用 REMOTE、LOCAL 进行配对，可以一对多、多对一、多对多
- 多个 KERNEL 的控制范围允许出现交集，比如分别控制贴图与颜色



用户导引

比较常用的应用

菜单形式

通过点击物体本体、linkmessage、gesture来唤起菜单，选择进行材质替换。

- 准备一个需要更换材质的目标物体，可以是perm、mesh、linkset
- 放入脚本
 - SMC.KERNEL
 - .SMC
 - SMC.Client
 - .SMC.Client
 - SMC.Menu
 - .SMC.Menu
- 撰写配置信息在 .SMC、.SMC.Client、.SMC.Menu
- 更改PRIM名称或备注
- (建议) **.SMC.Client**、**.SMC.Menu** 保存或者放入物体生效之后就可以删掉。
- (建议) 公频输入 **/finalise**，固化KERNEL的配置，此时可以删除 **.SMC**
- 点击物体开始使用

.SMC 与 .SMC.Client 中的 "LOCAL" 必须相同

HUD形式的应用

通过HUD与目标物体通信进行材质更换，远端控制。

- 准备一个物体，作为HUD
- 放入脚本
 - SMC.KERNEL
 - .SMC
 - (可选) **SMC.HUD.TRIGGER**，比较简单的HUD按钮点击触发器
 - 在HUD的按钮的备注中写入定义好的PART和SET，中间用 "." 分隔，比如 **PartA.Style1**。
SET 必须设置，PART 可以省略，如果不给予 PART 如: **.Style1**，将会替换包含SET为 **Style1** 的所有 PART。
 - 您可以开发HUD的触发脚本以实现更加丰富的操作，比如滑块、拾色器等。
- 撰写配置在 **.SMC**
- (建议) 公频输入 **/finalise**，固化KERNEL的配置，此时可以删除 **.SMC**
- 准备另一个需要更换材质的目标物体，可以是perm、mesh、linkset
- 放入脚本
 - SMC.Client
 - .SMC.Client
- (建议) 撰写配置在 **.SMC.Client**，这个文件保存或者放入物体之后就可以删掉了
- 重命名linkset中的prim
- 开始使用

.SMC 与 .SMC.Client 中的 "REMOTE" 必须相同

远端的菜单形式

也属于远端控制的一种，只是换成了菜单而不是去操作HUD。

- 准备好可触发菜单的物体
- 放入脚本
 - SMC.KERNEL
 - .SMC
 - SMC.Menu
 - .SMC.Menu
- 撰写配置信息在 .SMC、.SMC.Menu
- (建议) **.SMC.Menu** 保存或者放入物体生效之后就可以删掉。
- (建议) 公频输入 **/finalise**，固化KERNEL的配置，此时可以删除 **.SMC**
- 准备另一个需要更换材质的目标物体，可以是perm、mesh、linkset
- 放入脚本
 - SMC.Client
 - .SMC.Client
- (建议) 撰写配置在 **.SMC.Client**，这个文件保存或者放入物体之后就可以删掉了
- 重命名linkset中的prim
- 开始使用

.SMC 与 .SMC.Client 中的 "REMOTE" 必须相同

场景示例

一件衣服，有独立的HUD

- SMC.KERNEL 放在HUD里。
 - 可使用 SMC.HUD.TRIGGER，如果您有脚本基础，可以自行编写，更加灵活。
- SMC.Client 放在衣服里。
- SMC.KERNEL 与 SMC.Client 定义相同的 REMOTE。

一件衣服，点击领口弹出菜单

- SMC.KERNEL、SMC.Client、SMC.Menu 放在衣服里。
 - 放在 ROOT 或者 领口 都可以，取决于您想点击哪里弹出菜单。
- SMC.KERNEL 与 SMC.Client 定义相同的 LOCAL。

一栋房子，上面有个触控板，并且触控板与房子是一体的，点触控板弹出菜单

- SMC.KERNEL、SMC.Client 放在房子的任意 PRIM，定义相同的 LOCAL。
- SMC.Menu 放在触控板并开启 TOUCH。

一栋房子，上面有个触控板，并且触控板与房子是分体的，点触控板弹出菜单

- SMC.Client 放在房子的任意 PRIM。
- SMC.KERNEL、SMC.Menu 放在触控板并开启 TOUCH。
- SMC.KERNEL 与 SMC.Client 定义相同的 REMOTE。

一栋房子，上面有2个触控板，触控板有与房子一体的，还有个分体的，点击弹出菜单。您兜里还有一个，能当HUD用

- SMC.Client、SMC.KERNEL、SMC.Menu 放在房子的任意 PRIM，SMC.Menu 开启 TOUCH
- SMC.KERNEL、SMC.Menu 放在分体的触控板，SMC.Menu 开启 TOUCH
- SMC.KERNEL、SMC.Menu 放在您兜里的触控板HUD。
- 房子里的 SMC.Client、SMC.KERNEL 定义相同的 LOCAL。
- 分体的和您兜里的 SMC.KERNEL 定义与房子里 SMC.Client 相同的 REMOTE。

注意！**SMC.HUD.TRIGGER** 只能用于包含独立 **PRIM** 按钮的 **HUD**，它依赖于不同的名称或者备注。如果只有一个 **PRIM**，是不行的，它**无法**对面甚至面的触摸位置 (**ST/UV**) 进行识别。如果需要，您可以自行撰写。

配置

.SMC

| 配置项 | 类型 | 取值 | 默认 | 说明 |
|--------|---------|---------------------------------|----|--|
| DEBUG | integer | 0 / 1 | 0 | 排障模式，开启时会输出较多信息 |
| LOCAL | integer | -2147483648 ~ 2147483647 (0 无效) | 0 | 本地通信频道，多用于菜单形式 |
| REMOTE | integer | -10000 ~ 10000 | 0 | 远程通信频道偏移量（注意: 这是私有频道偏移量，并不是确切的频道），一般用于HUD。 |

| 配置项 | 类型 | 取值 | 默认 | 说明 |
|-------|---------|---------|----|--|
| CACHE | integer | 0/1 | 0 | 资源缓冲(UUID), 如果配置中使用的图片出现大量重用的情况, 建议开启, 可以节省大量内存。 |
| RANGE | integer | 0/1/2/3 | 0 | 控制距离, 0:10米, 1:20米, 2:100米, 3:整个地区 |
| LINES | list | | | 详细书写规则会在下文中介绍 |

LINES

PART

部位/目标/选择器

- PART代表一个或者多个目标(prim + face), 会被更换材质的部位。可以理解为: PART所定义的是一个选择(查找)器。
- PART后面必须跟随**4个参数**

```
list LINES = [  
    PART, "{名称}", {匹配类型}, "{匹配文本}", {面}  
];
```

| 参数 | 类型 | 取值 | 说明 |
|------|----------------|--|--|
| 名称 | string | 任意 | 一组LINES的配置中, 不可重复, 这是用来换材质的依据之一, 在本地菜单模式中也会作为选项来使用 |
| 匹配类型 | integer | 见下表 | 用于描述匹配的类型 |
| 匹配文本 | string/integer | 用于匹配的名称或者描述, 与参数2配合进行定义 | |
| 面 | string/integer | 目标PRIM的哪个(些)面, PRIM的面编号(0~7)。可以传递字符串比如"0267", 将会匹配多个面, 不必按顺序, 但不可重复。也可以写 ALL_SIDES(-1), 此时不可再写其它面, 因为ALL_SIDES代表所有面。 | |

匹配类型

| 常量 | 对应值 | 说明 |
|---------|-----|---|
| FULL | 0 | 全文匹配PRIM名称 |
| PREFIX | 1 | 匹配PRIM名称的前缀 |
| SUFFIX | 2 | 匹配PRIM名称的后缀 |
| SMART | 3 | 智能匹配PRIM名称(暂时不可用) |
| CONST | 4 | 匹配文本以SL中常量的方式，匹配文本可以是: LINK_SET, LINK_ALL_CHILDREN, LINK_ALL_OTHERS, LINK_ROOT, LINK_THIS |
| DFULL | 10 | 全文匹配PRIM描述 |
| DPREFIX | 11 | 匹配PRIM描述的前缀 |
| DSUFFIX | 12 | 匹配PRIM描述的后缀 |
| DSMART | 13 | 智能匹配PRIM描述(暂时不可用) |

举例

匹配名称为 "A" 的PRIM的第 3、4 面

```
list LINES = [  
  PART, "Part A", FULL, "A", "34"  
];
```

匹配 "名称前缀是 Rect" 的PRIM的 所有 面

```
list LINES = [  
  PART, "All part starting with Rect", PREFIX, "Rect", ALL_SIDES  
];
```

匹配 "名称末尾是 3" 的PRIM的第 0 面

```
list LINES = [  
  PART, "All part ending with 3", SUFFIX, "3", 0  
];
```

匹配 "除了脚本所在PRIM之外的其他PRIM" 的第 1、2、5 面

```
list LINES = [
  PART, "All others", CONST, LINK_ALL_OTHERS, "125"
];
```

匹配 "备注前缀为 top" 的第 所有 面

```
list LINES = [
  PART, "TOP", DSUFFIX, "top", ALL_SIDES
];
```

SET

配色/主题/材质方案

- SET代表着一套材质方案，它是非常自由的配置方式。
- SET的定义不能独立存在，它必须跟在一个PART后面。
- SET对应多种属性，属性所需参数数量也会有所不同。

```
list LINES = [
  PART, ...,
  SET, {属性}, ..., {属性}, ..., {属性}, ...
];
```

属性

参考 [PRIM_TEXTURE](#)

| 属性 | 对应值 | 对应属性 | 描述 | 参数数量 | 值 | 说明 |
|----|-----|------------------------------|----------|------|----------------------------|--------------|
| D | 0 | PRIM_TEXTURE | 漫反射贴图 | 1 | "{UUID}" | 仅换图，其它属性继承 |
| DP | 1 | PRIM_TEXTURE | 漫反射 (详细) | 4 | "{UUID}", {重复}, {位置}, {旋转} | 设置漫反射相关的所有属性 |
| N | 2 | PRIM_NORMAL | 硬表面贴图 | 1 | "{UUID}" | 仅换图，其它属性继承 |

| 属性 | 对应值 | 对应属性 | 描述 | 参数数量 | 值 | 说明 |
|----|-----|-----------------|----------|------|--|-------------------------------|
| NP | 3 | PRIM_NORMAL | 硬表面 (详细) | 4 | "{UUID}", {重复}, {位置}, {旋转} | 设置硬表面相关的所有属性 |
| S | 4 | PRIM_SPECULAR | 光泽贴图 | 1 | "{UUID}" | 仅换图，其它属性继承 |
| SP | 5 | PRIM_SPECULAR | 光泽 (详细) | 7 | "{UUID}", {重复}, {位置}, {旋转}, {反光颜色}, {反光度}, {环境光强度} | 设置光泽相关的所有属性 |
| C | 6 | PRIM_COLOR | 颜色 | 1 | {颜色} | color 与 alpha 可以分开设置 |
| A | 7 | PRIM_COLOR | 透明度 | 1 | {透明度} | color 与 alpha 可以分开设置 |
| G | 8 | PRIM_GLOW | 发光 | 1 | {强度} | 灯泡一样的光 |
| F | 9 | PRIM_FULLBRIGHT | 全亮模式 | 1 | {TRUE/FALSE} | 开启或者关闭 |
| B | 10 | PRIM_BUMP_SHINY | 硬表面和反光 | 2 | {强度}, {模式} | 系统自带的那个 |
| T | 11 | PRIM_TEXGEN | 贴图模式 | 1 | {模式} | 默认/平面 |
| M | 12 | PRIM_ALPHA_MODE | 透明模式 | 2 | {模式}, {遮罩屏蔽} | 不管用不用 alpha masking，第二个参数都不能少 |

如果值给予空字符串，表示不替换（继承当前的值）

举例

更换贴图，全部硬表面，透明度，发光

```
list LINES = [  
    PART, ...,  
    SET, "name_1", D, "{uuid}", NP, "{uuid}", <1.0, 1.0, 0.0>, <0.0, 0.0, 0.0>, 0.0,  
    A, 0.6, G, 0.02  
]
```


更换颜色，全亮模式，清空光泽贴图

```
list LINES = [
  PART, ...,
  SET, "name_2", C, <1.0, 0.0, 0.0>, F, TRUE, S, NULL_KEY
]
```

更换漫反射的位置和旋转，同时，不改变现有的贴图和重复

```
list LINES = [
  PART, ...,
  SET, "name_3", DP, "", "", <0.125, 0.4, 0.0>, 135.65
]
```

.SMC.Client

| 配置项 | 类型 | 取值 | 默认 | 说明 |
|----------|---------|---------------------------------|-----|--|
| DEBUG | integer | 0 / 1 | 0 | 排障模式，开启时会输出较多信息 |
| LOCAL | integer | -2147483648 ~ 2147483647 (0 无效) | 0 | 本地通信频道，多用于菜单形式 |
| REMOTE | integer | -10000 ~ 10000 | 0 | 远程通信频道偏移量（注意: 这是私有频道偏移量，并不是确切的频道），一般用于HUD。 |
| DEBOUNCE | float | ≥ 0.0 | 0.0 | 防抖时长，在这个时间内的变化均会累计，直到没有更换材质的操作并在此时长后开始生效，避免频繁切换带来的效率瓶颈。 |
| CACHE | integer | 0 / 1 | 0 | 选择器缓存，用缓存换取更高效的匹配速度，注意：开启本选项后，不可以对物体进行link与unlink操作，否则会出现错误。 |

.SMC.Menu

| 配置项 | 类型 | 取值 | 默认 | 说明 |
|------------|---------|-------|----|--|
| DEBUG | integer | 0 / 1 | 0 | 排障模式，开启时会输出较多信息 |
| TOUCH | integer | 0 / 1 | 0 | 菜单可否由点击弹出 |
| OWNER_ONLY | integer | 0 / 1 | 0 | 点击着是否必须为所有者 |
| SETS | integer | 0 / 1 | 0 | 套装选项，在部位（PART）列表中增加"[SETS]"选项，进入套装（SETS）列表菜单 |

| 配置项 | 类型 | 取值 | 默认 | 说明 |
|---------------------|---------|---------------------------------|------|--|
| SETS_ON_TOP | integer | 0 / 1 | 0 | 顶级菜单 部位 (PART) 列表 被替换为 套装 (SETS) 列表 |
| PARTS | integer | 0 / 1 | 0 | 如果 SETS_ON_TOP 开启, 在 套装 (SETS) 菜单中增加一项 "[PART]", 作为部位 (PART) 菜单的入口 |
| MENU_OPEN_LOCAL_NUM | integer | -2147483648 ~ 2147483647 (0 无效) | 0 | 触发菜单弹出的本地 num |
| MENU_BACK_LOCAL_NUM | integer | -2147483648 ~ 2147483647 (0 无效) | 0 | 返回上一层菜单的回调 |
| MENU_BACK_OVERWRITE | string | 任意 | 空字符串 | 替换返回选项文案 |
| MENU_PREV_OVERWRITE | string | 任意 | 空字符串 | 替换前一页选项文案 |
| MENU_NEXT_OVERWRITE | string | 任意 | 空字符串 | 替换后一页选项文案 |
| SETS_LIST | integer | 0 / 1 | 0 | 见下文样例 |

SETS_LIST

格式

```
list SETS_LIST = [
    "{套装名称}", "{PART}.{SET}",
    ...
];

list SETS_LIST = [
    "{套装名称}", ".{SET}",
    ...
];

list SETS_LIST = [
    "{套装名称}", ".{SET_A},{SET_B},{PART1}.{SET_C},...",
    ...
];
```

```
...  
];
```

举例

```
list SETS_LIST = [  
    "BLACK", ".BLACK"  
];
```

```
list SETS_LIST = [  
    "BLACK&RED", ".BLACK,.RED"  
];
```

```
list SETS_LIST = [  
    "BLACK&TOP_RED", ".BLACK,TOP.RED"  
];
```

```
list SETS_LIST = [  
    "BTM_B&T_R", "BOTTOM.BLACK,TOP.RED"  
];
```

核心 KERNEL 本地接口

消息字符串分隔符是 "◆"

```
llDumpList2String([...], "◆")
```

提交

-643323390

对预定义部位应用一个预定义属性，并支持自定义追加与覆盖

```
llMessageLinked(LINK_SET, -643323390, "{PART}◆{SET}[/◆{DATA...}]", "");
```

- PART 与 SET 必须在配置中定义过，另外 SET 必须属于 PART，本条提交才会生效。
- DATA 部分为追加或覆盖属性，写法如 SET 中的属性，可选参数。

举例

```
// 最常用的(使用预定义配置 LINES)
llMessageLinked(LINK_SET, -643323390, "TOP◆BLACK", "");
// 带有自定义属性的
llMessageLinked(LINK_SET, -643323390, "TOP◆BLACK◆6◆<1.0, 0.0,
0.0>◆9◆TRUE◆4◆ee509dfd-0974-6fb5-3eea-2504fa13ef4c", "");
// 方便的写法
llMessageLinked(LINK_SET, -643323390, llDumpList2String(["TOP", "BLACK", 6, <1.0,
0.0, 0.0>, 9, TRUE, 4, "ee509dfd-0974-6fb5-3eea-2504fa13ef4c"], "◆"), "");
// 建议使用常量, 可写为
llMessageLinked(LINK_SET, -643323390, llDumpList2String(["TOP", "BLACK", C, <1.0,
0.0, 0.0>, F, TRUE, S, "ee509dfd-0974-6fb5-3eea-2504fa13ef4c"], "◆"), "");
```

* 批量模式

```
llMessageLinked(LINK_SET, -643323390, "◆{SET}", "");
```

- 如果不给予 PART, 此时将触发全量匹配模式, 自动查找出包含 SET 的所有 PART, 并批量生效。
- 此时再追加的 DATA, 它们会应用在所有相关 PART。

举例

```
// 定义的 PART 中, 比如有 TOP、MIDDLE、BOTTOM
// 其中 TOP、MIDDLE 包含 BLACK, 那么, 将会自动查找出 TOP、MIDDLE, 应用 BLACK
// 相当于执行了 TOP◆BLACK 和 MIDDLE◆BLACK
llMessageLinked(LINK_SET, -643323390, "◆BLACK", "");
```

-643323392

对预定义部位应用一套自定义属性

```
llMessageLinked(LINK_SET, -643323392, "{PART}◆{DATA...}", "");
```

- PART 必须在配置中定义过, 本条提交才会生效。
- DATA 的写法如 SET 中的属性, 对 PART 应用自定义属性, 与上面的区别是无需指定 SET (不验证 SET 的合法性)。

举例

```
llMessageLinked(LINK_SET, -643323392, "TOP◆6◆<1.0, 0.0,
0.0>◆9◆TRUE◆4◆ee509dfd-0974-6fb5-3eea-2504fa13ef4c", "");
// 方便的写法
llMessageLinked(LINK_SET, -643323392, llDumpList2String(["TOP", 6, <1.0, 0.0,
0.0>, 9, TRUE, 4, "ee509dfd-0974-6fb5-3eea-2504fa13ef4c"], "◆"), "");
// 建议使用常量, 可写为
```

```
llMessageLinked(LINK_SET, -643323392, llDumpList2String(["TOP", C, <1.0, 0.0, 0.0>, F, TRUE, S, "ee509dfd-0974-6fb5-3eea-2504fa13ef4c"], "💎"), "");
```

-643323393

对自定义的部位应用一套自定义属性

```
llMessageLinked(LINK_SET, -643323393, "{DATA...}", "");
```

- DATA 的写法必须包含完整的 PART + SET 内容。
- 无需依照配置，这是一条完全独立的选择 + 属性规则。

举例

```
llMessageLinked(LINK_SET, -643323393, "2💎top💎0123💎6💎<1.0, 0.0, 0.0>💎9💎TRUE💎4💎ee509dfd-0974-6fb5-3eea-2504fa13ef4c", "");
// 方便的写法
llMessageLinked(LINK_SET, -643323393, llDumpList2String([2, "top", "0123", 6, <1.0, 0.0, 0.0>, 9, TRUE, 4, "ee509dfd-0974-6fb5-3eea-2504fa13ef4c"], "💎"), "");
// 建议使用常量, 可写为
llMessageLinked(LINK_SET, -643323393, llDumpList2String([SUFFIX, "top", "0123", C, <1.0, 0.0, 0.0>, F, TRUE, S, "ee509dfd-0974-6fb5-3eea-2504fa13ef4c"], "💎"), "");
```

请求(带回调)

-643323410

请求 PART 列表

```
llMessageLinked(LINK_SET, -643323410, "", id);
```

KERNEL 回调: -643323411

```
llMessageLinked({SENDER}, -643323411, "{PART1}💎{PART2}💎....", id);
```

-643323420

请求 SET 列表

```
llMessageLinked(LINK_SET, -643323420, "{SET}", id);
```

KERNEL 回调: -643323411

```
llMessageLinked({SENDER}, -643323421, "{SET1}◆{SET2}◆....", id);
```