# 图块自动化参数说明



## 目标

图层自动化命令。

## 使用

### 命令

打开终端，输入如下指令：

```
layerauto {map_file}
```

该指令读入地图路径，将转换结果自动输到原路径。

### 命令参数

```
-o --output 输出到不同路径（建议，防止覆盖后无法悔改）。

-v --verbose 显示运行信息

--ignorewarning 发生warning时，会继续执行，而不会退出程序。

-y --isyes 自动同意程序的一切(y/n)请求

--language 后仅允许跟"ch"(中文)/"eg"(英文)。语言设置将会被储存，之后使用延续上一次的修改。

-j --config 图层自动化设置的json文件。后面可以添加多个json文件，会依次执行。
```

**前置知识**

图层是如何存储信息并显示地图的呢？图层实际上是一个数字表格。每一个数字表示一个地块。这个数字被翻译成哪一个地块需要看该地图载入了哪些地块集。因此，绘制底图本质上来说是修改这一个数字表格，使得合适的位置写上合适的数字。地图中会出现多个图层 (Ground, Items, Units 等等)，这些图层各自都是数字表格。图层自动化就可以使用一些设置来对这些数字表格进行有规律的处理，从而减少绘制底图过程中的重复性工作。

然而，宾语层和图层的组织方式不同。每一个宾语都是一个独立的单位，这个宾语记载了它的上下左右边界、名称类型以及属性等，这些决定了这个宾语。有时候我们希望一个特定的宾语也可以对其位置的地形造成影响。因此，我们需要将宾语层”翻译”为正常的图层。哪些宾语可以被映射，怎么映射，映射为哪一个数字，也需要设置来决定。

现在，图层和宾语虚拟层 (被映射为数字表格的宾语层) 就可以执行图层自动化了。我们这里得到了若干相同大小的数字表格。我们希望做一些什么事情来自动改变图块呢?

这不是一个宾语自动化系统，针对宾语虚拟层的改变不会影响宾语层，只能拿来做图层自动化的参数或者说标记。

1. 映射: 某些地层的地块满足某些条件时，将会对其他地层的地块造成影响。
2. 地形:

(1) 自动平铺/转角集。自动平铺可以改变地形，并且保证地形边缘保持自然。
(2) 河流, 铁路, 道路/边缘集。边缘集可以使得河流等地绘制方向保持正确自然。

3. 扩大: 希望城市宾语对周围的地块都造成影响。

为了达成以上目标，这样一个算法被设计了出来。我们可以通过产生新的执行虚拟层 (类似宾语虚拟层)，并在这个虚拟层上运算，并将这个虚拟层按照一定的规则映射到图层。如此，我们可以满足这些要求。

**自动化步骤**

1. 宾语层生成一个空的相同名字宾语虚拟层，名称符合正则表达式的宾语将在对应坐标赋值。(可以不执行)
2. 将会执行若干次，形成执行虚拟层。

(1) 首先根据要求将目前的层和虚拟层映射进新形成的执行虚拟层。
(2) 执行虚拟层将会执行运算指令。(可以不执行)
(3) 执行虚拟层映射赋值改变其他的层或虚拟层。(可以不执行)
(4) 重复执行 (1),(2),(3)，直到结束。

**json 格式**

```
1     {
2         "rwmapauto_type": "layerauto",
```

```
    "simplify": {},
    "objectre_to_layer": {},
    "execution": []
}
```

json 文件中有四项内容，第一项是默认的”rwmapauto_type”：“layerauto”来确保是用于该命令的 json 文件(可选)。

第二项是”simplify”字典(可选)。键为简化的地块集名称，值为实际地块集名称(与地块集的名称一样)。之后就可以使用简化地块集名称表示这些地块集了(可选)。值有两种，一种是普通的字符串，需要完全匹配地块集或图层。第二种是一个列表，第一个是一个正则表达式，第二个必须是”re”，表明这是正则表达式。这样将会模糊匹配地块集或图层。

```
{
    "simplify": {
        "tile_T": ["^.*巴巴罗萨计划(?!.*渐变海洋).*地块byXs.*$", "re"],
        "tile_S": ["^.*巴巴罗萨计划.*渐变海洋地块byXs.*$", "re"],
        "tile_ass": ["^辅助地块.*1.2.*byXs.*$", "re"],
        "layer_Ground": "Ground",
        "layer_Items1": "Items",
        "layer_Items2": "ItemsExtra"
    }
}
```

这样之后用”tile_T”表示巴巴罗萨地块。

第三项是”objectre_to_layer”字典(可选)。该字典用于图层自动化步骤 1。该字典可以为空，那么将不会进行宾语层映射。键为宾语层名称(需要映射的宾语层，一般仅有 Triggers)，值为一个列表。列表内存储着字典。每一个字典表示了一种映射模式。其中字典内”re”表示当宾语的名称符合哪些正则表达式时，该宾语将会被映射。字典内的”map_type”表示映射模式。目前有两种模式：“middle”表示在该宾语的中心点映射到虚拟层，“left-top”表示在该宾语的左上角映射到宾语虚拟层，默认为”left-top”(可选)。字典内的”gid”表示可以映射的值，应当为大于 0 的整数，因为层初始化均为 0。列表内的映射模式会依次执行。

```
{
    "objectre_to_layer": {
        "Triggers":[
            {
                "re": "^.*c\\..*$",
                "map_type": "middle",
                "gid": 1
            }
        ]
    }
}
```

第四层是”execution”列表。是图层自动化最核心的部分，是图层自动化步骤 2 的参数。列表内存储着若干次生成执行虚拟层参数的字典。

在图层自动化步骤 2 进行之前，我们通过宾语层映射得到了宾语虚拟层以及已有的图层。我们生成的执行虚拟层均是基

于当前拥有的所有层。执行虚拟层生成后，下一步生成也可以使用之前执行虚拟层的数据。

下面是对生成执行虚拟层的字典的描述。“exe_name”对应了生成执行虚拟层的名称，类似于”Ground”等。“layer_to_exe”是一个列表，内部存储了 2.(1) 步骤的信息，代表层对执行虚拟层的映射。“exe”是一个列表，列表内存储着字典，内部存储了 2.(2) 步骤的信息，表示执行虚拟层的内部运算。“exe_to_layer”是一个字典，内部存储了 2.(3) 信息，表示执行虚拟层对层的映射。关于后三项的内容后面具体解释。

```
{
    "execution": [
        {
            "exe_name": "city",
            "layer_to_exe": [],
            "exe": [],
            "exe_to_layer": {}
        }
    ]
}
```

“layer_to_exe”是一个列表。这个列表内存储的是一个长度为 2 的列表。列表内第一个元素是一个字典，表示哪些层符号哪些条件时进行映射。第二个元素是一个正整数，表示映射的数字。对于第一个字典来说，键为层的名称，值为满足的条件。当字典内所有条件均满足时，执行虚拟层将会被映射一个数字。将所有映射模式从前往后执行，从而得到映射的执行虚拟层。这个满足的条件有很多格式。仅有一个数字是用于宾语虚拟层和执行虚拟层的条件。如果是一个字典，那么字典的键是地块集的名称 (可被 simplify 简化)，值有两种模式。值是一个长度为 2 的列表。列表内每一个元素如果都是整数，则代表是这个地块集内该坐标的地块。列表内每一个元素如果都是长度为 2 的列表，那么就是一个矩阵，范围是第一个坐标到第二个坐标框起来的区域。坐标从 0 开始，从上到下是第一个元素，从左到右是第二个元素。数字或字典可以被列表整合，表示该地层的该地块满足的条件范围。

```
{
    "execution": [
        {
            "layer_to_exe": [
                [{"Triggers": 1}, 1]
                [{"Ground": [{"tile_T": [5, 12]}, {"tile_S": [[1, 0],
                    [6, 11]]}]}, 1]
            ]
        }
    ]
}
```

意思是，Triggers 宾语虚拟层为 1 或者 Ground 层为”tile_T”地块集 (5, 12) 或”tile_S”地块集 (1, 0) 到 (6, 11) 的矩形时映射执行层为 1。

“exe”是一个列表。列表内有若干字典。执行虚拟层将会依次执行这些字典内的命令。字典内的”exe_type”是执行虚拟层操作的类型。目前共有三种类型。

“expansion”：“exe_operation”是一个字典，是一个数字字符串映射到 3×3 的数字列表的字典。将会按照键从小到大顺序执行这一”扩展”操作，原地执行。如果检测到执行虚拟层某一个数字与键相同，将会把其以及附近 3×3 范围

进行赋值，按照这一数字列表进行赋值。如果执行虚拟层赋值区域不是 0(非空) 或者数字列表值是-1(不进行赋值)，那么赋值失败，否则赋值成功。

```
{
    "execution": [
        {
            "exe": [
                {
                    "exe_type": "expansion",
                    "exe_operation": {
                        "1": [
                            [-1,  2, -1],
                            [ 2, -1,  2],
                            [-1,  2, -1]
                        ],
                        "2": [
                            [-1,  3, -1],
                            [ 3, -1,  3],
                            [-1,  3, -1]
                        ],
                        "3": [
                            [-1,  4, -1],
                            [ 4, -1,  4],
                            [-1,  4, -1]
                        ]
                    }
                }
            ]
        }
    ]
}
```

这代表着，一旦执行虚拟层有 1。那么其将会向外扩展为

```
      4
    4 3 4
  4 3 2 3 4
4 3 2 1 2 3 4
  4 3 2 3 4
    4 3 4
      4
```

“terrain”：“exe_operation”是一个字典，是一个数字字符串映射到 3×3 的数字列表的字典。将会对符合所有 3×3 列表模式的位置赋值该数字 (键)，新生成数字表格赋值。“-1”表示不在意这一位置是什么数字。否则要求匹配。

特别的，可以添加一个额外选项”exe_border”。这将需要一个字典。这个字典表明该地形处理过程中，地图边缘应该怎么处理。目前有一个选项”all”。“all”内的数字表明地图边缘将会被认为是什么数字。-1 表明无论如何都通过。-2 表明无论如何都不通过。

```
{
```

```
    "exe": [
        {
            "exe": [
                {
                    "exe_type": "terrain",
                    "exe_border": {
                        "all": 1
                    },
                    "exe_operation": {
                        "11111": [
                            [ -1,  1,  -1],
                            [ 1,  1,  1],
                            [ -1,  1,  -1]
                        ],
                        "11110": [
                            [ -1,  1,  -1],
                            [ 1,  1,  1],
                            [ -1,  0,  -1]
                        ],
                        "11101": [
                            [ -1,  1,  -1],
                            [ 1,  1,  0],
                            [ -1,  1,  -1]
                        ],
                        "11100": [
                            [ -1,  1,  -1],
                            [ 1,  1,  0],
                            [ -1,  0,  -1]
                        ],
                        "11011": [
                            [ -1,  1,  -1],
                            [ 0,  1,  1],
                            [ -1,  1,  -1]
                        ],
                        "11010": [
                            [ -1,  1,  -1],
                            [ 0,  1,  1],
                            [ -1,  0,  -1]
                        ],
                        "11001": [
                            [ -1,  1,  -1],
                            [ 0,  1,  0],
                            [ -1,  1,  -1]
                        ],
                        "11000": [
                            [ -1,  1,  -1],
                            [ 0,  1,  0],
                            [ -1,  0,  -1]
                        ],
                        "10111": [
                            [ -1,  0,  -1],
```

```
                                [ 1,  1,  1],
                                [ -1,  1,  -1]
                            ],
                            "10110": [
                                [ -1,  0,  -1],
                                [ 1,  1,  1],
                                [ -1,  0,  -1]
                            ],
                            "10101": [
                                [ -1,  0,  -1],
                                [ 1,  1,  0],
                                [ -1,  1,  -1]
                            ],
                            "10100": [
                                [ -1,  0,  -1],
                                [ 1,  1,  0],
                                [ -1,  0,  -1]
                            ],
                            "10011": [
                                [ -1,  0,  -1],
                                [ 0,  1,  1],
                                [ -1,  1,  -1]
                            ],
                            "10010": [
                                [ -1,  0,  -1],
                                [ 0,  1,  1],
                                [ -1,  0,  -1]
                            ],
                            "10001": [
                                [ -1,  0,  -1],
                                [ 0,  1,  0],
                                [ -1,  1,  -1]
                            ],
                            "10000": [
                                [ -1,  0,  -1],
                                [ 0,  1,  0],
                                [ -1,  0,  -1]
                            ]
                        }
                    }
                ]
            }
        ]
}
```

这是河流或者铁路等的执行。

“exe_to_layer” 是一个字典，表示执行虚拟层将哪些数字映射到哪些层的图块。键为执行虚拟层的数字。值是一个列表，列表内每一个元素都是 4 个元素的列表。这四个元素依次是映射的层 (1)，地块集名称 (2)，地块集坐标 (3, 4)。意思是将在映射的层的对应位置修改为地块集的对应坐标的图块。可以加多个 4 个元素列表，表示对多个图层均进行映

射。特别的，“empty”被认为是一个空地块集，后面必须跟 0,0，表示该位置被设置为空。

```
{
    "exe": [
        {
            "exe_to_layer": {
                "5": [["ItemsExtra", "empty", 0, 0]],
                "1": [["ItemsExtra", "tile_T", 2, 13], ["
                    PathingOverride", "tile_ass", 9, 4]]
            }
        }
    ]
}
```

执行虚拟层为 5 时，ItemsExtra 对应位置为空。执行虚拟层为 1 时，ItemsExtra 对应位置为”tile_T”地块集的 (2, 13) 地块，“PathingOverride”对应位置为”tile_ass”地块集的 (9, 4) 地块。

json 文件介绍完毕。通过合适的设置，就可以得到有效的图层自动化模式并提高制作地图的效率。