随机元素分配

编写者：刘品资

编写日期：2020-2-5

**修改记录：**

|  |  |  |
| --- | --- | --- |
| 修改人 | 修改日期 | 修改内容 |
| 刘品资 | 2020-2-5 | 初稿 |
|  |  |  |

# 单层随机元素的分配问题（分配在模块和模块房间）

## 问题：单层元素A随机得到数量X后，如何分配在各个模块和房间比较合理。（假设单层有Y个模块，总的房间数量为Z）

## 方案1：

* 通过平均分配房间分配：每个房间至多X/Z+1个随机元素A。
* 所以每个房间的上限为X/Z+1。
* 随机将元素往各个房间丢，某房间到上限了就不会再往该房间丢。确定房间多少个该元素后，比如3个，但这个房间只有2个位置能放该元素，再去找个同模块内有位置的房间给它塞进去(此时将不管上限)。塞不了的就废弃掉。

PS：上限值的结果统一向上取整。

## 结论：该方案下，是将元素相对平均的随机分配在每个房间。如果房间之间相差较大，则随机下来后可能会不太合理。

## 方案2：

通过房间内的触发区域数量来分类定义房间小、中、大三个级别。

小房间[0-8]，中房间[9-15]，大房间[16，正无穷]（后续可更改）

再给各类房间设定小中大的上下限：设小房间a个（可分配元素占比20%），中房间b个（可分配元素占比35%），大房间c个（可分配元素占比45%）。

则小房间的上限：[20%\*a/(20%\*a+35%\*b+45%\*c)]/a

则中房间的上限：[35%\*b/(20%\*a+35%\*b+45%\*c)]/b

则大房间的上限：[45%\*c/(20%\*a+35%\*b+45%\*c)]/c

PS：上限值的结果统一向上取整。

## 结论：该方案下，通过定义小中大三种房间。将元素按照房间级别随机分配在每个房间。相比方案1，会根据房间的大小来决定分配多少元素的上限，不用担心房间相差过大所产生的问题。

## 方案3：

设任意房间内的触发区域数为d，单层总触发区域数为D。

则任意房间的上限为d/D\*X。

PS：上限值的结果统一向上取整。

## 结论：该方案下，直接通过每个房间的触发区域数来决定该房间会分配到多少元素数量的上限，相比1、2方案更加精准，且不用通过区域数来定义小中大房间及其占比

# 编辑器配置数据结构

## 门的数据配置：

* 门的类型：服务端会根据拥有入口门和出口门的房间中随机一个房间为入口房间和出口房间（其中出入口门是可以随机为入口门或出口门）

1. 模块内部连接门
2. 模块的入口门
3. 模块的出口门
4. 模块的出入口门

* 所属房间ID：
* 门的坐标：
* 门的资源目录：

## 固定元素的数据配置：

* 所属房间ID：
* 坐标点：
* 事件ID等相关配置：

## 随机元素的数据配置：

* 所属房间ID：
* 坐标点：
* 可随机事件ID等相关配置：