详 细 设 计 文 档

——冬津游艺

## 详细设计目录

[1.引言 2](#_Toc11580)

[1.1编写目的 2](#_Toc30782)

[1.2项目背景 2](#_Toc14231)

[1.3术语说明 2](#_Toc32289)

[1.4参考资料 4](#_Toc11184)

[2.软件结构 5](#_Toc12978)

[2.1 需求概述 5](#_Toc29968)

[2.2软件结构 5](#_Toc21627)

[3.程序设计说明 6](#_Toc31895)

[3.1模块描述 6](#_Toc17239)

[3.2功能 8](#_Toc15182)

[3.3性能 12](#_Toc4729)

[3.4输入项 13](#_Toc9399)

[3.5 输出项 13](#_Toc26609)

[3.6算法 13](#_Toc11476)

[3.6.1游戏逻辑模块.刷新模块：魔物刷新算法 13](#_Toc32036)

[3.6.2 游戏控制模块：游戏物体控制算法 16](#_Toc3903)

[3.7 程序逻辑 17](#_Toc28671)

[3.7.1 游戏逻辑模块.刷新模块：魔物刷新算法 17](#_Toc26619)

[3.8 接口 18](#_Toc21249)

[3.9测试要点 19](#_Toc29682)

# 1.引言

## 1.1编写目的

本文目的是明确说明系统需求，界定系统实现功能的范围，指导系统设计以及编码。使本系统的使用者和软件开发者双方对该软件的初始规定有一个共同的理解。

本文描述了《冬津游艺》游戏的主要功能设计，分析了本项目在技术、可玩性条件方面的需求，主要叙述了系统的功能和非功能方面的设想，明确了本项目的目标和工作计划。

本报告的预期读者是项目组长、设计人员、开发人员、测试人员、本系统的最终使用者。

## 1.2项目背景

冬津游艺游戏软件：

委托单位：中软国际

负责单位：opssw组成员

主管部门：西北工业大学软件学院

用户：玩家

## 1.3术语说明

|  |  |  |
| --- | --- | --- |
| **序号** | **术语** | **所指对象或含义** |
| **1** | **元素** | **在游戏软件游玩过程中有七种元素，分别称作：水、火、雷、冰、岩、风、草。魔物具有一项状态，该状态记录是否挂有七种元素中的一种，元素间可以两两发生反应** |
| **2** | **反应** | **当魔物已经挂有元素A时，再次对魔物挂有元素B，则元素A和元素B便会反应，消除魔物的挂有元素的状态并产生特殊效果（11~23）** |
| **3** | **植物** | **用于阻止魔物来到最左端的对象，由操作的玩家放置在地图上，可进行自动攻击对魔物造成伤害，伤害值转换为减少魔物的生命值，攻击可带有七种元素之一，会使收到攻击的魔物挂有该元素** |
| **4** | **魔物** | **魔物会从地图最右端向左移动，玩家必须阻止这些单位进入地图的最左端，否则游戏失败**  **魔物会持续攻击行进时遇到的植物，对植物造成伤害，伤害值转换为减少植物的生命值，期间会停止向左移动，直到该植物单位生命值为零** |
| **5** | **能量** | **植物需要一定的能量才能放置在地图上，可随游戏运行时间增加或由特定的植物产生** |
| **6** | **生命值** | **植物与魔物的共同属性，值为零时倒下，从地图上消除** |
| **7** | **物理攻击** | **没有元素属性的攻击** |
| **8** | **抗性** | **魔物对某一个元素的攻击存在抗性。例如，对于雷元素拥有70%抗性的魔物，能够抵御70%的雷元素伤害。100点对此魔物的雷元素伤害仅会造成30点伤害。** |
| **9** | **韧性** | **魔物的韧性度量值。当魔物受到高于此值的冲击时，魔物会产生停滞动作，停止前进一段时间。** |
| **10** | **冲击** | **植物的攻击会对魔物产生冲击，当冲击值大于魔物韧性值时，魔物会产生停滞动作** |
| **11** | **冻结** | **水+冰产生的元素反应。会使目标进入一段时间的冻结状态，移动变慢。在持续时间里，目标受到冰元素附着。当目标附着的冰元素产生了其他反应时，目标会解冻；当目标受到了物理伤害攻击时，目标会解冻并受到额外伤害，此伤害为固定值。目标解冻后，移除冻结效果。** |
| **12** | **蒸发** | **水+火产生的元素反应。触发这次反应的伤害会提高至1.75倍。** |
| **13** | **融化** | **冰+火产生的元素反应。触发这次反应的伤害会提高至1.75倍。** |
| **14** | **超导** | **雷+冰产生的元素反应。造成小范围伤害并减少目标的物理抗性。** |
| **15** | **超载** | **雷+火产生的元素反应。造成中范围伤害并对魔物产生冲击。** |
| **16** | **感电** | **雷+水产生的元素反应。造成小范围持续伤害。在持续时间里，目标仍受到水、雷元素附着。当在感电伤害未结算完成前再次发生除感电以外的元素反应，则感电效果立即结束，水雷元素消失，并正常结算新元素反应。** |
| **17** | **绽放** | **草+水产生的元素反应。在魔物脚下产生一个草原核，短暂时间后会爆炸造成范围伤害，此伤害为固定值。当草原核受到了雷元素的伤害时，还会产生超绽放反应，当草原核受到了火元素的伤害时，还会产生烈绽放反应** |
| **18** | **超绽放** | **将会产生覆盖地图的追踪弹，追踪攻击魔物。** |
| **19** | **烈绽放** | **将会在大范围造成范围伤害。** |
| **20** | **激化** | **草+雷产生的元素反应。造成伤害同时减少魔物的草属性抗性和雷属性抗性各30%** |
| **21** | **燃烧** | **草+火产生的元素反应。对魔物持续造成伤害。在持续时间里，目标会受到火元素附着。此火元素附着不会随着元素反应的发生而消失，直至持续时间结束。** |
| **22** | **扩散** | **风+（冰/水/火/雷）产生的元素反应。消除魔物挂有的元素，对魔物造成消除元素的元素伤害，并为魔物小范围内其他魔物挂有被扩散元素。** |
| **23** | **结晶** | **岩+（冰/水/火/雷）产生的元素反应。造成额外伤害并掉落结晶块。结晶块可被小范围内的植物吸收产生一层临时的护盾，护盾可减少魔物对植物的伤害。** |

## 1.4参考资料

1. 软件工程导论 张海藩著 清华大学出版社2013.08
2. 植物大战僵尸，美国宝开游戏公司开发的益智策略类塔防御战游戏
3. Genshin Impact，[上海米哈游网络科技股份有限公司](https://baike.baidu.com/item/%E4%B8%8A%E6%B5%B7%E7%B1%B3%E5%93%88%E6%B8%B8%E7%BD%91%E7%BB%9C%E7%A7%91%E6%8A%80%E8%82%A1%E4%BB%BD%E6%9C%89%E9%99%90%E5%85%AC%E5%8F%B8/19987087?fromModule=lemma_inlink)制作发行的[开放世界](https://baike.baidu.com/item/%E5%BC%80%E6%94%BE%E4%B8%96%E7%95%8C/15488017?fromModule=lemma_inlink)冒险游戏

[4] 冬津游艺需求规格文档

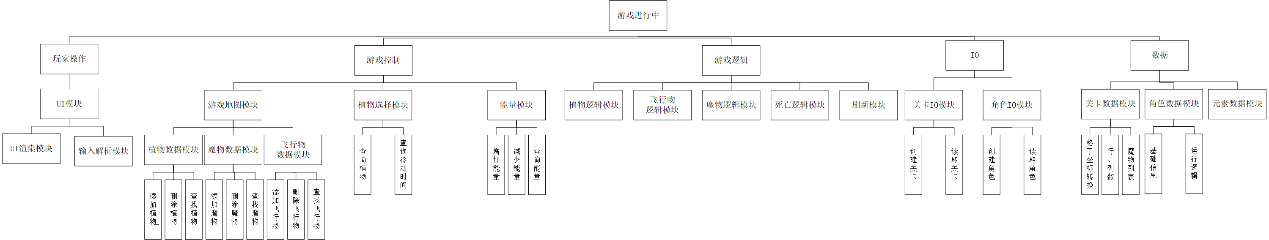
[5] 冬津游艺概要设计文档

# 2.软件结构

## 2.1 需求概述

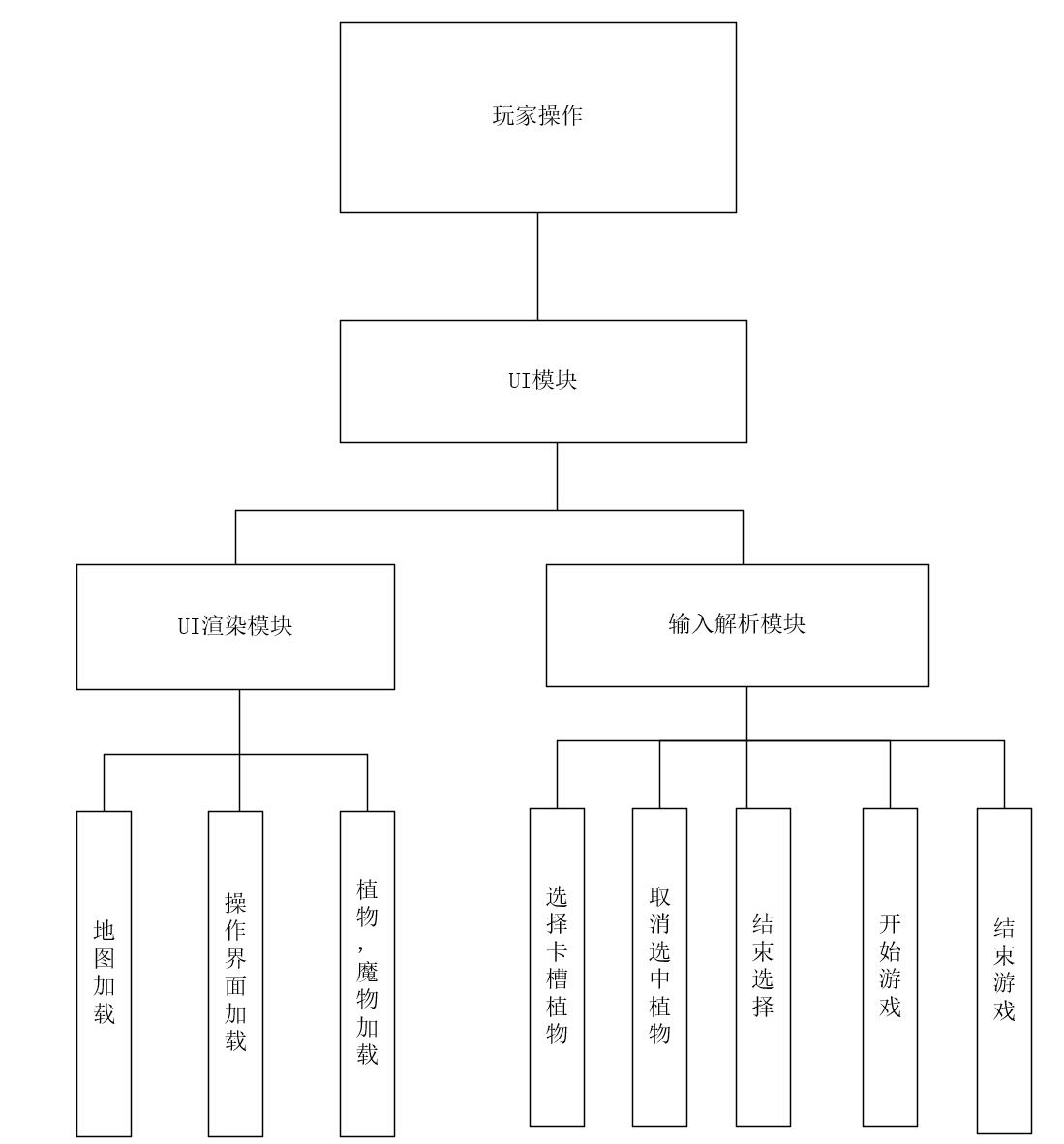
冬津游艺是一款益智类塔防游戏，游戏机制借鉴于植物大战僵尸，玩家可在游玩的过程中，通过收集元素能量等材料来部署不同的植物角色，切换不同的功能，在地图上来抵抗魔物的进攻，守卫提瓦特大陆。不同的敌人，不同的玩法构成不同的游戏模式，其中灵感出自原神的元素反应将作为游戏游玩的核心，以提高策略性与挑战性。

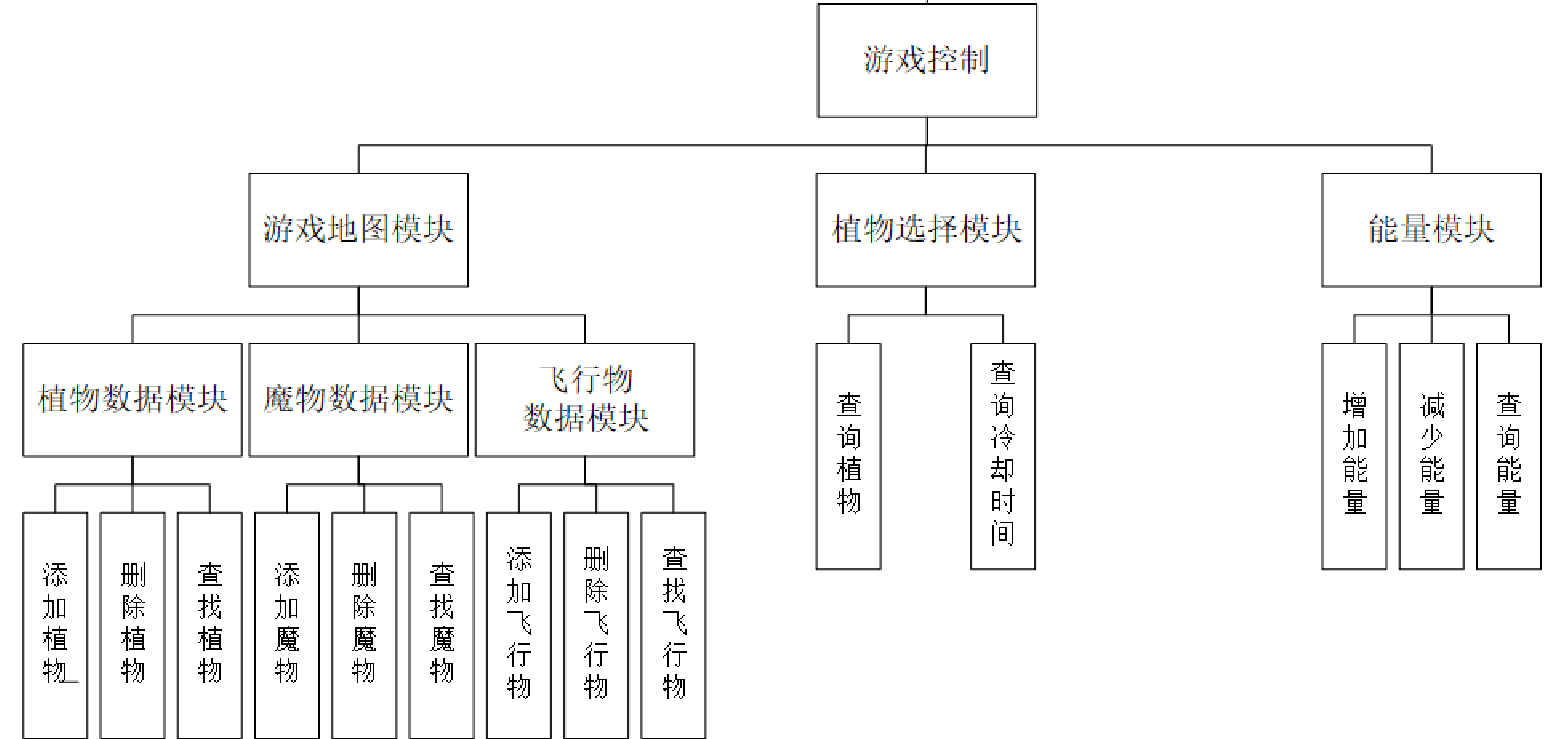
## 2.2软件结构

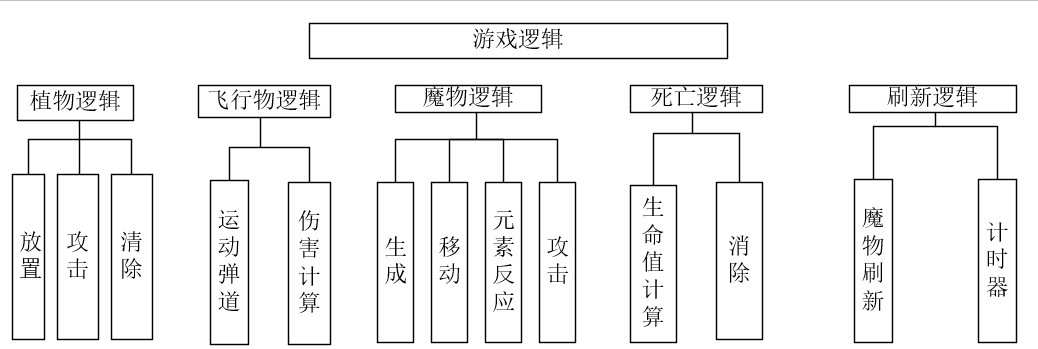


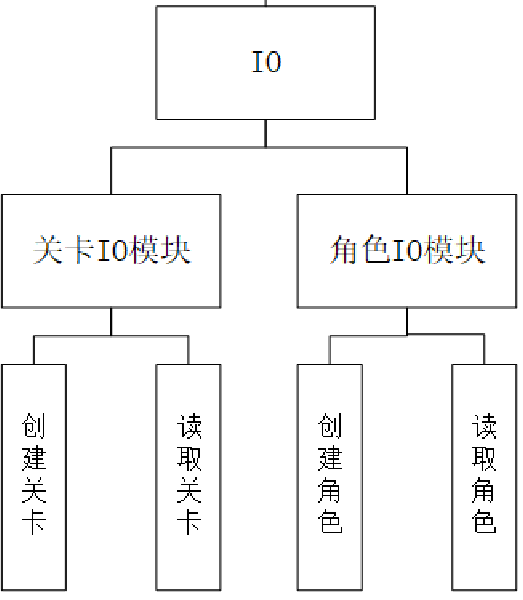
# 3.程序设计说明

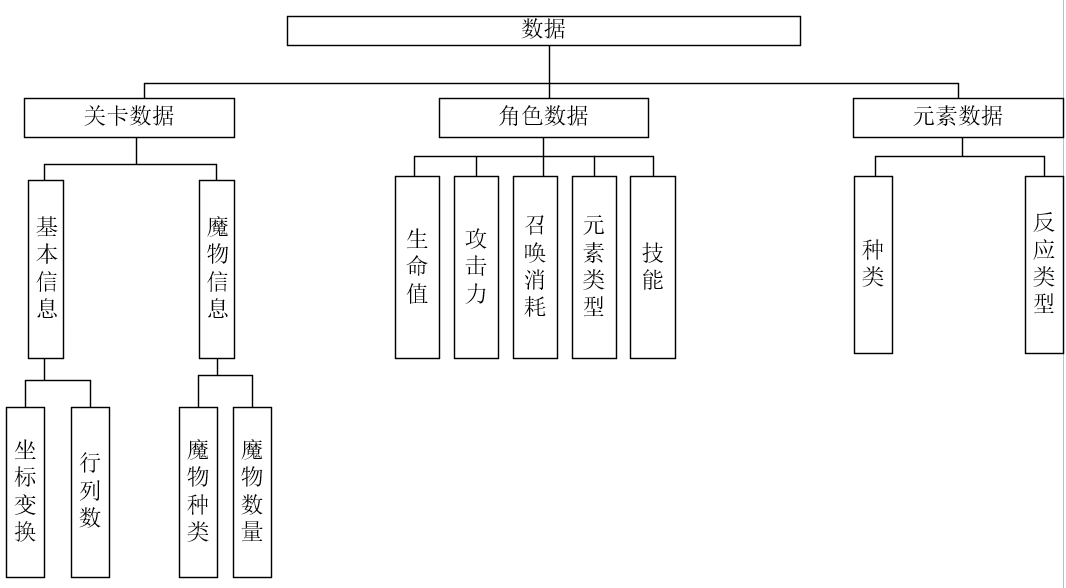
## 3.1模块描述





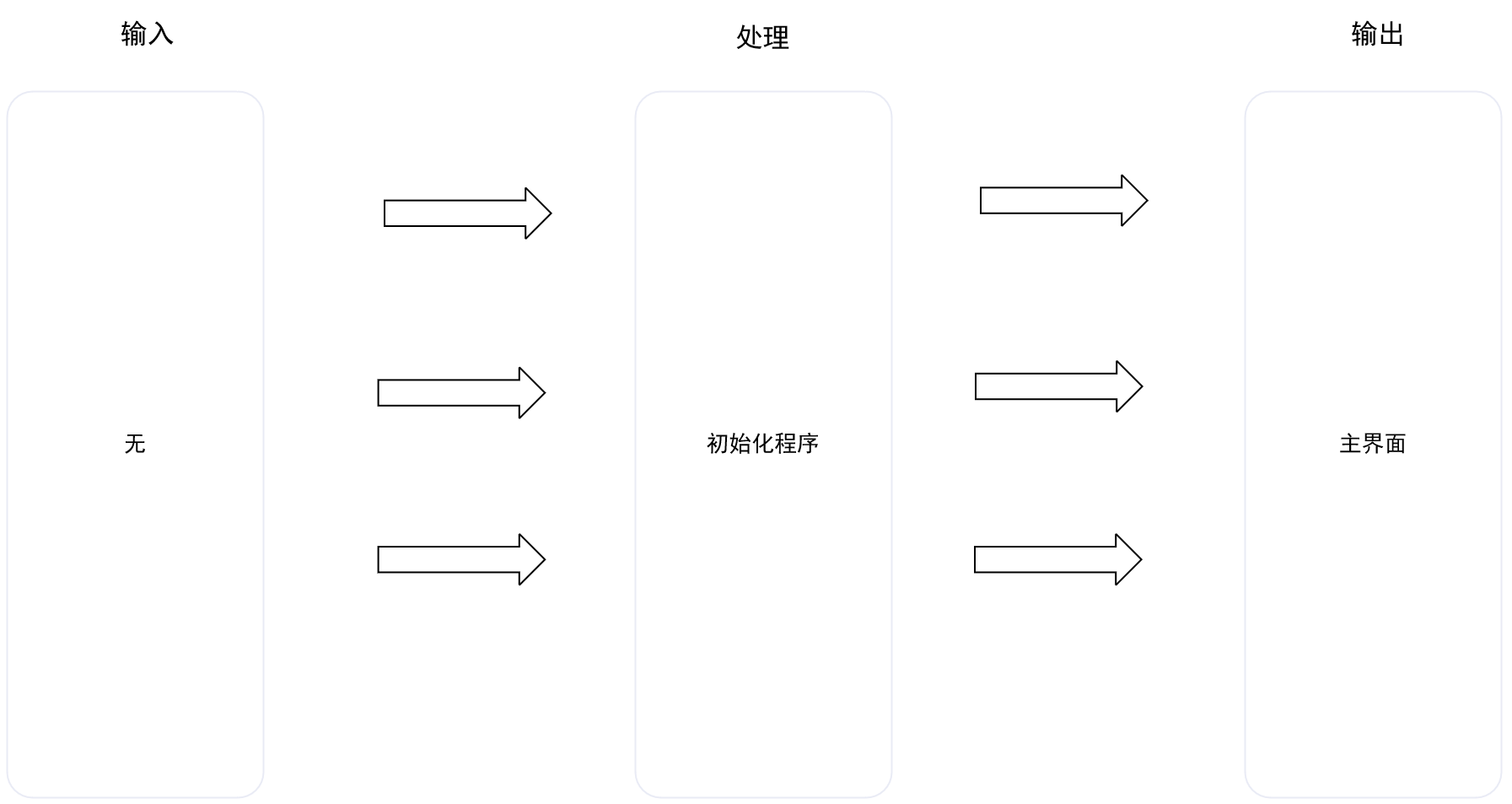




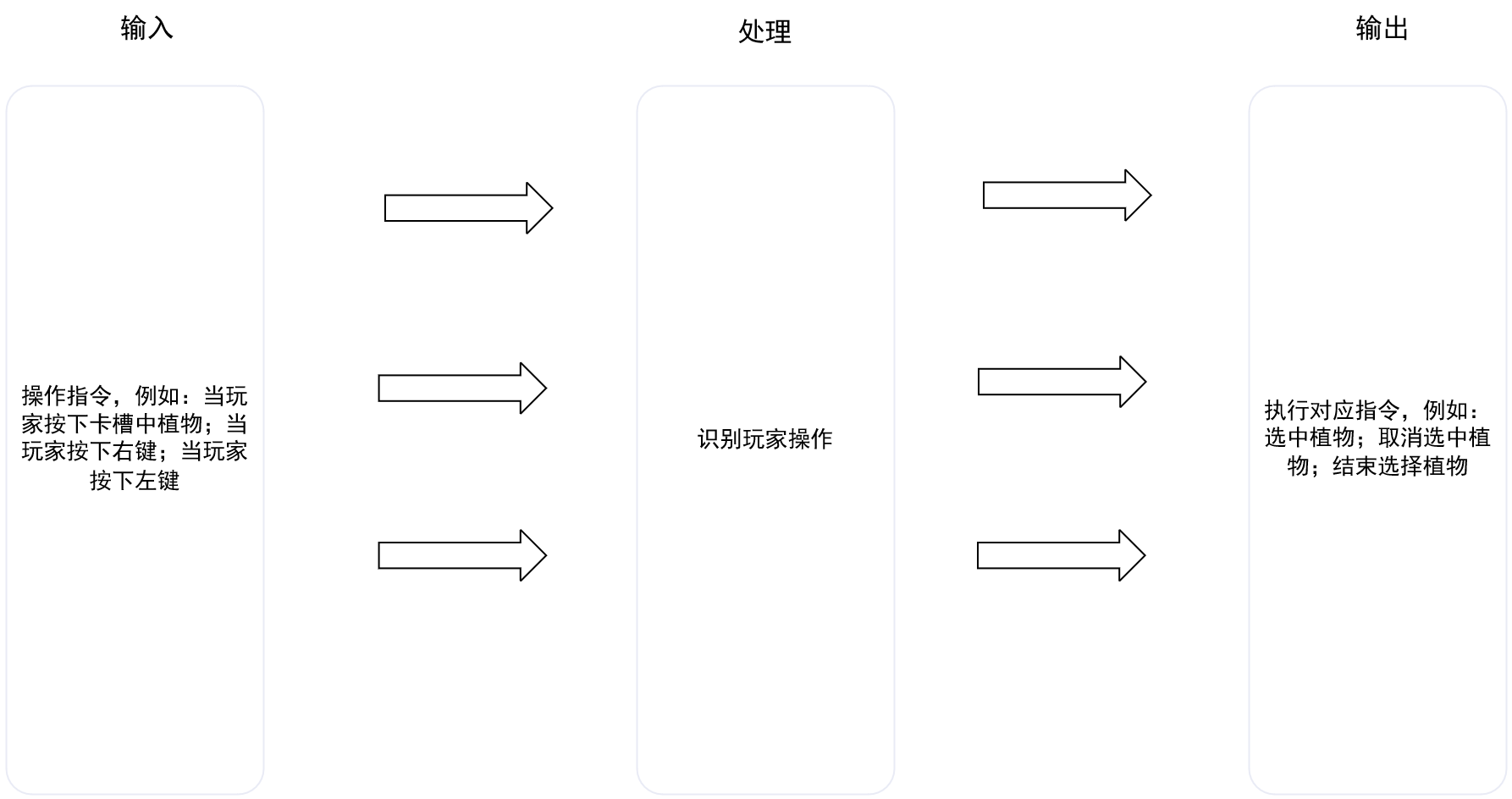


## 3.2功能

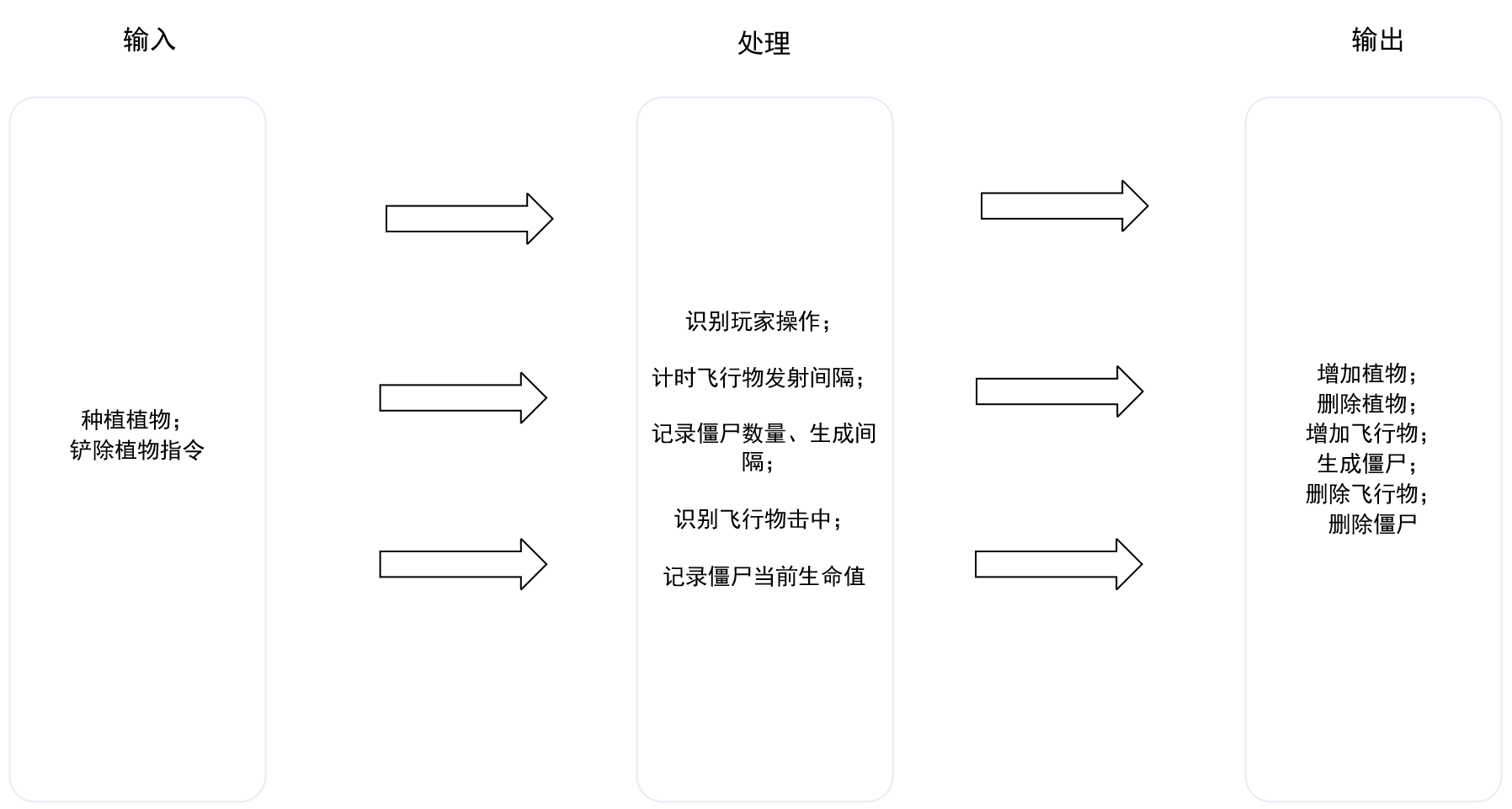
1. UI渲染模块



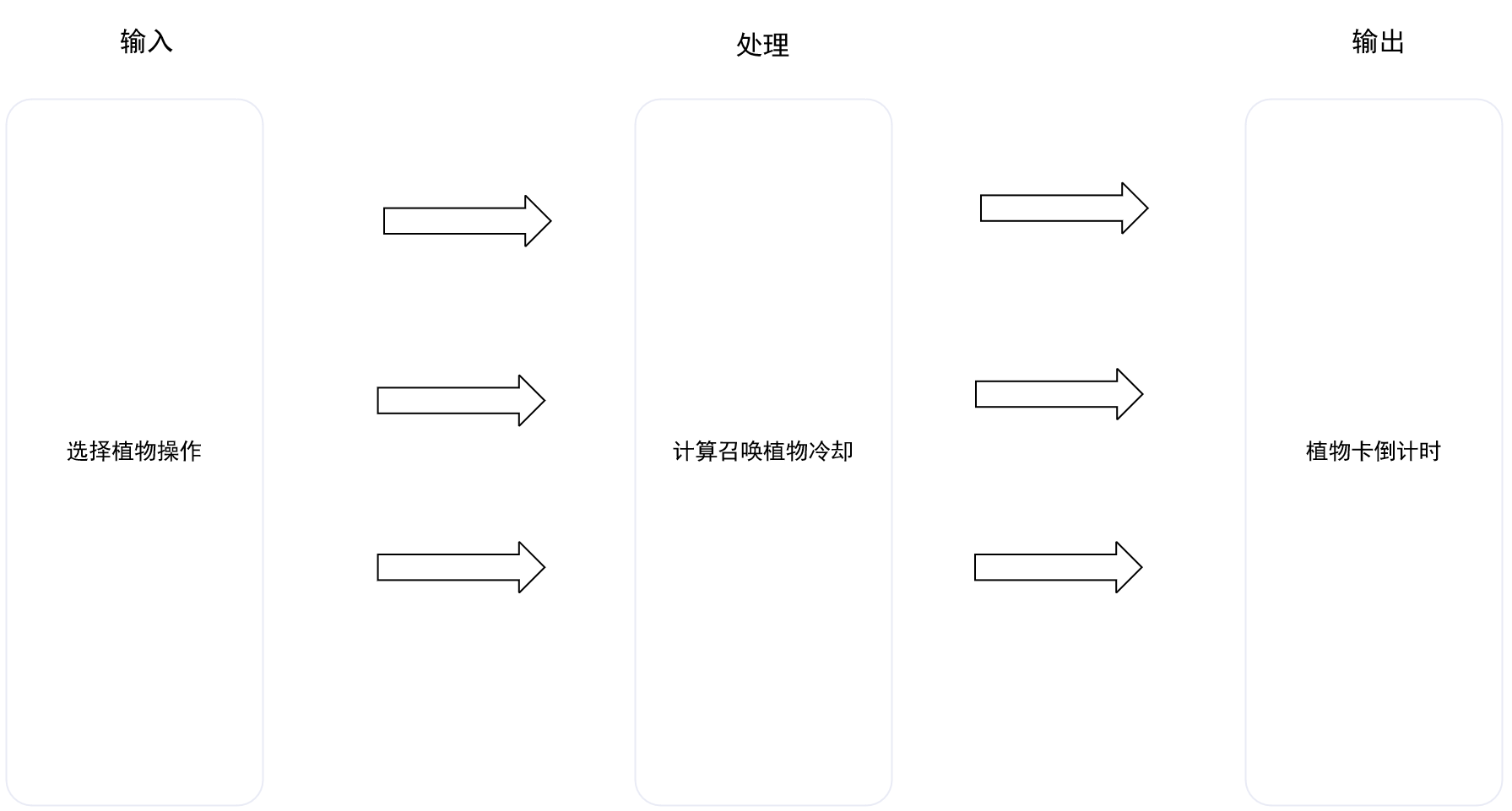
1. 输入解析模块



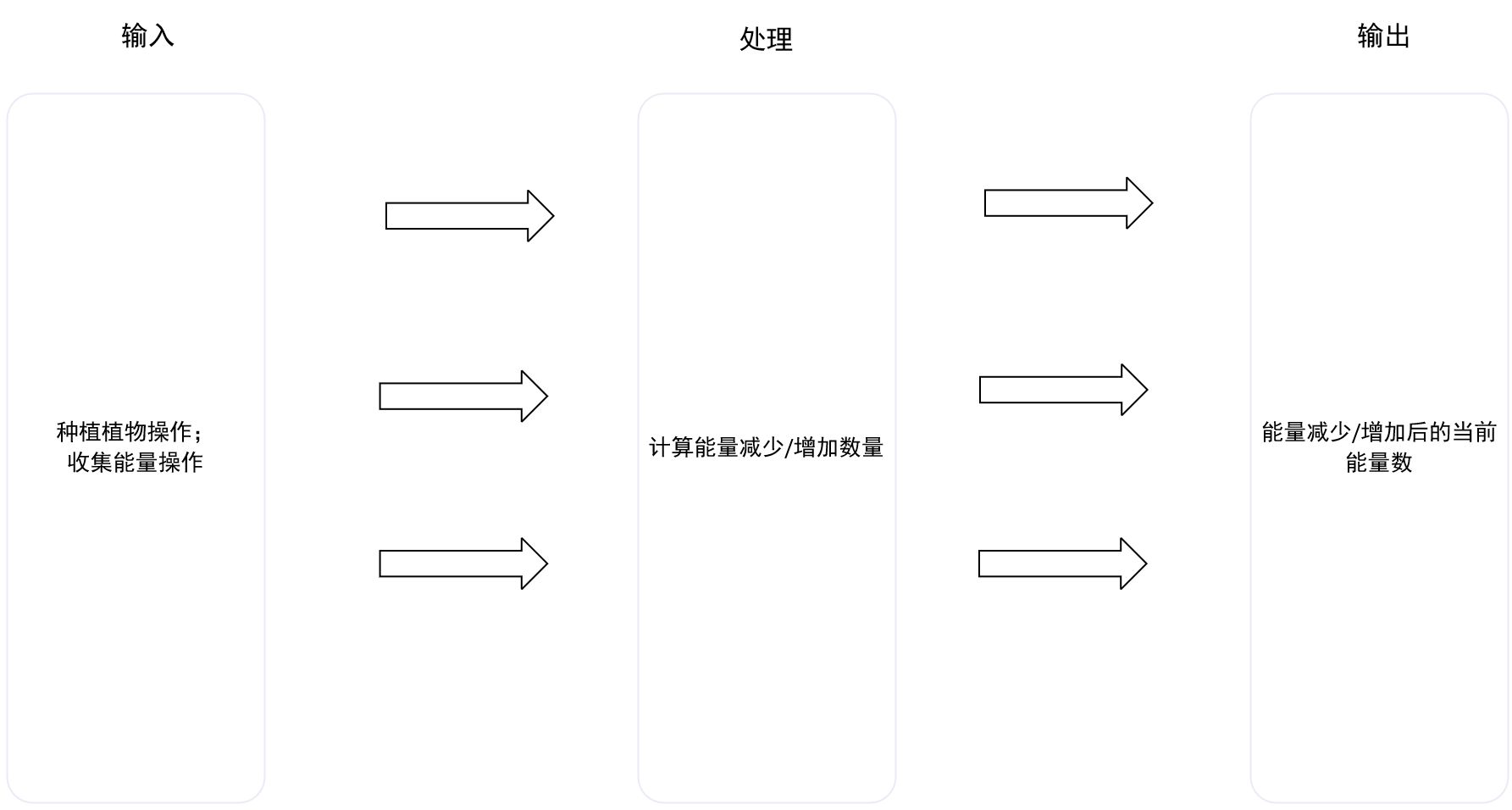
1. 游戏控制模块



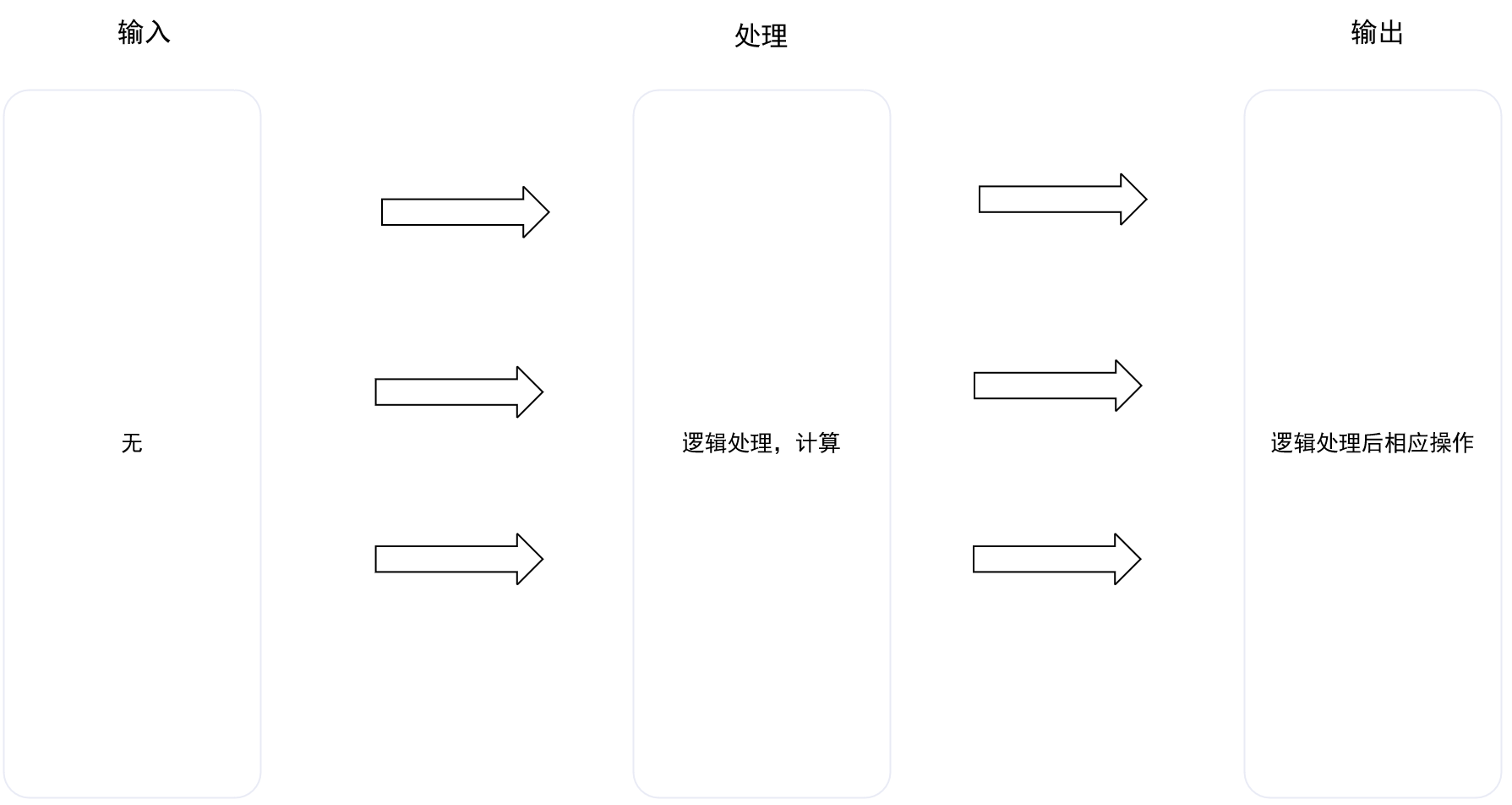
1. 植物选择模块



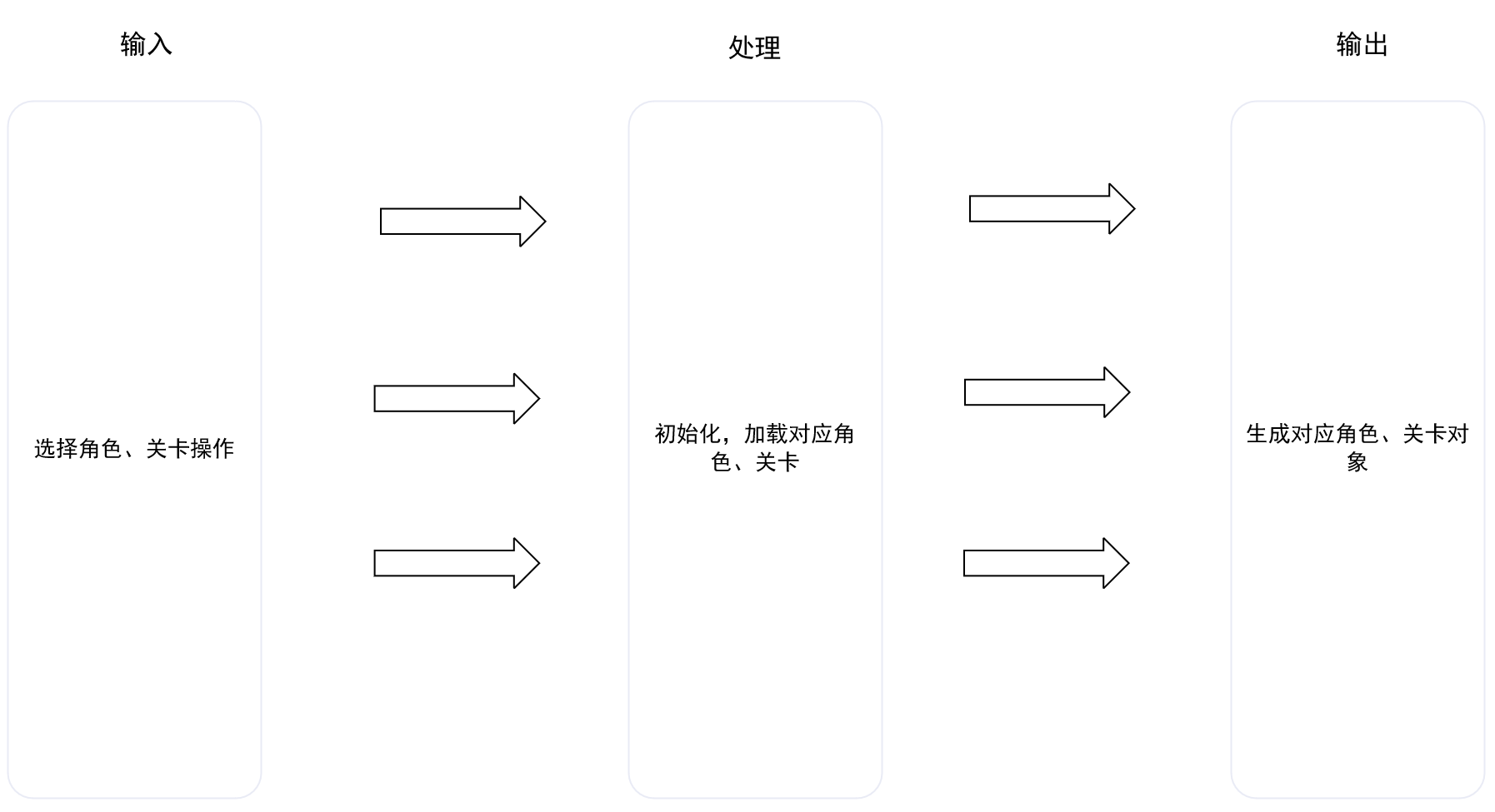
1. 能量模块



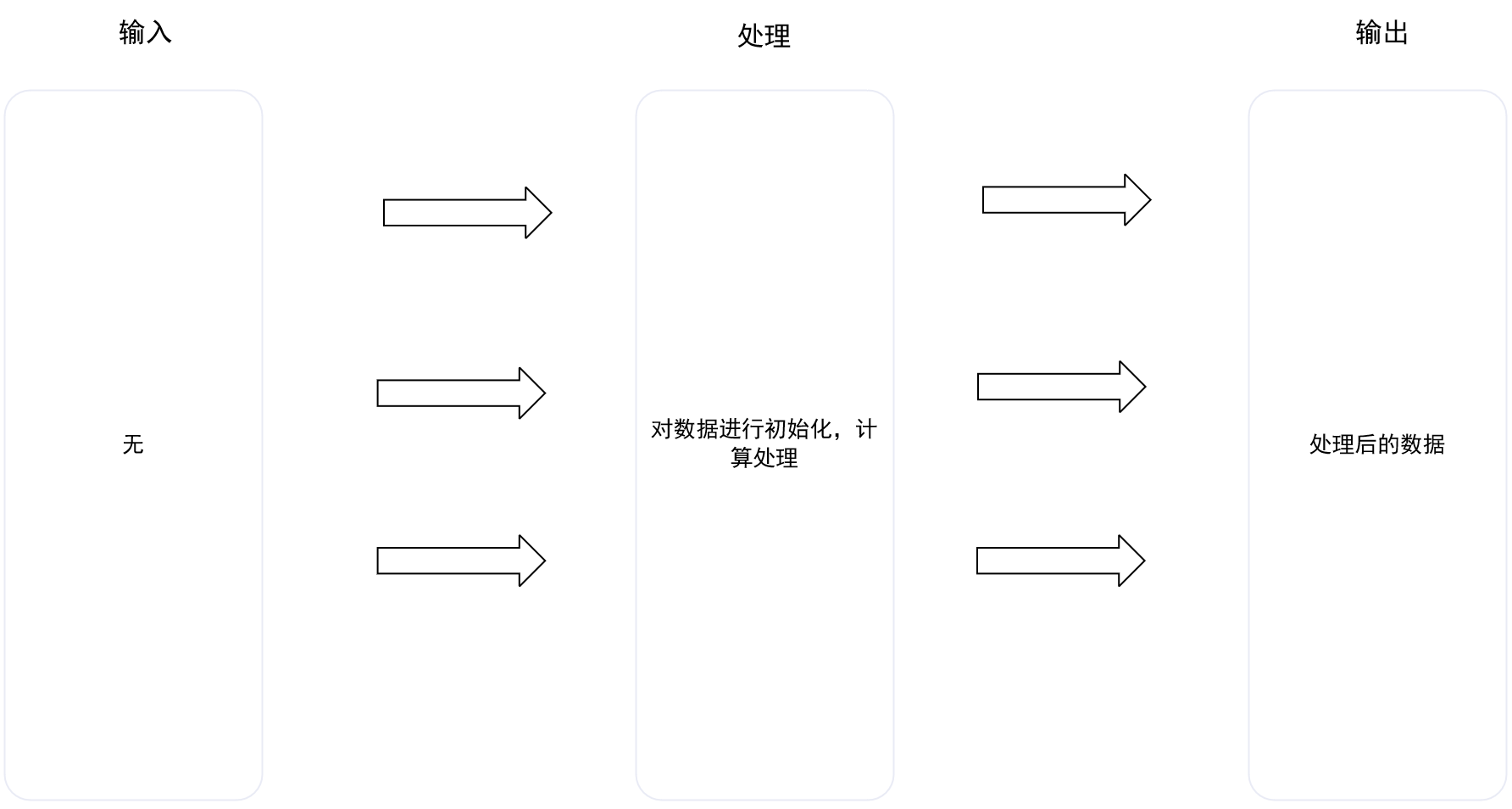
1. 逻辑模块



1. IO模块



1. 数据模块



## 3.3性能

UI渲染、IO、输入解析、游戏地图，能量，逻辑模块要求程序有较快的响应速度，建议反应时间不超过100ms；数据模块要求精度精确到整数位；逻辑模块要求程序有较好灵活性，能够及时正确处理错误信息

## 3.4输入项

### 3.4.1 UI渲染模块

UI元素位置：float:Left,float:Top,float:Bottom,float:Right（锚框型）和float:PosX,float:PosY,float:Width,float:Height（锚点型）

### 3.4.2输入解析模块

鼠标位置(float:PosX,float:PosY)、鼠标点击事件(Event)、鼠标拖拽事件(Event)

### 3.4.3游戏控制模块

植物、魔物、飞行物的数据，游戏开始触发(Event)，游戏结果(boolean)

植物数据: 卡牌贴图Sprite,游戏物体原型GameObject,能量消耗int,冷却时间int,生命值int,攻击力int,植物名字string,植物介绍string

魔物数据:游戏物体原型GameObject,生命值int,攻击力int,速度int, 韧性int

飞行物数据：游戏物体原型GameObject

### 3.4.4植物选择模块

植物数据: 卡牌贴图Sprite,游戏物体原型GameObject,能量消耗int,冷却时间int,生命值int,攻击力int,植物名字string,植物介绍string

### 3.4.5能量模块

能量新值int，能量增加值int，能量减少值int

### 3.4.6游戏逻辑模块

此模块为帧更新模块，大部分数据由自己或游戏控制维护，由unity帧更新函数调用而并无输入项。

### 3.4.7 IO模块

关卡名字string、植物名字string、飞行物名字string、魔物名字string、音乐名字string、音效名字string

### 3.4.8 数据模块

数据模块的初始值在构造函数中就已经写定，并无输入项。

## 3.5 输出项

### 3.5.1 UI渲染模块

UI元素显示：整个屏幕的像素颜色数组。

### 3.5.2输入解析模块

对应事件执行(Event)

### 3.5.3游戏控制模块

植物、魔物、飞行物的游戏实体(GameObject)，游戏结果显示执行（Event）

### 3.5.4植物选择模块

植物当前冷却时间int

植物数据: 卡牌贴图Sprite,游戏物体原型GameObject,能量消耗int,冷却时间int,生命值int,攻击力int,植物名字string,植物介绍string

### 3.5.5能量模块

能量值变化监听函数（Event）

### 3.5.6游戏逻辑模块

游戏逻辑下一帧内容。包括植物的效果触发、飞行物移动触发、魔物移动和身上效果分析、魔物刷新等。（Event）

### 3.5.7 IO模块

关卡数据、植物数据、飞行物数据、魔物数据、音乐音效数据

关卡数据：关卡图片Sprite，行数int，列数int，起始能量值int

植物数据: 卡牌贴图Sprite,游戏物体原型GameObject,能量消耗int,冷却时间int,生命值int,攻击力int,植物名字string,植物介绍string

魔物数据:游戏物体原型GameObject,生命值int,攻击力int,速度int, 韧性int

飞行物数据：游戏物体原型GameObject

音乐音效数据：音乐切片AudioClip

### 3.5.8 数据模块

此类对象的新实例。

## 3.6算法

### 3.6.1游戏逻辑模块.刷新模块：魔物刷新算法

依托PVZ游戏原版的刷新算法，此算法实现了普通出怪和大规模出怪两种模式。根据游戏进度情况，在游戏前期会刷出较少、较弱的魔物；在游戏后期会刷出较多、较强的魔物。

游戏目前实现的魔物种类包含普通僵尸、路障僵尸、铁桶僵尸。设计上来说，魔物的强度大小排列为：铁桶僵尸>路障僵尸>普通僵尸。在算法设计中，引入了以下概念。

1. 魔物出现总数：此关卡中出现的魔物总数量。
2. 当前出现数量：目前已经生成的魔物数量。
3. 关卡进度：标识着目前关卡的进度。一般以**（2）**当前出现数量 /**（1）**魔物出现总数 作为其值大小。其大小是0~1之间的浮点数。
4. 魔物出现间隔：魔物并非随时刷新，而是每次刷新结束间隔一段时间才会再次刷新。
5. 魔物真实最小出怪量：魔物的最小出怪量，最终的最小出怪量会得到修正值的补偿。
6. 魔物真实最大出怪量：魔物的最大出怪量，最终的最大出怪量会得到修正值的补偿。
7. 魔物出怪修正值：随着**（3）**关卡进度的增加，魔物的出现量也会越来越多。其值相当于当**（3）**关卡进度达到1时，最终最小出怪量和最终最大出怪量相比于真实最小出怪量和真实最大出怪量，所增加的数量值。
8. 魔物最终最小出怪量：魔物的最终最小出怪量。
9. 魔物最终最大出怪量：魔物的最终最大出怪量。当场上怪物大于此值时，刷新算法会停止刷怪，直到场上的怪物小于此值。
10. 尝试生成数量：随机整数，大小在**（8）**魔物最终最小出怪量 和**（9）**魔物最终最大出怪量之间。得出尝试生成数量后开始根据权值和概率生成魔物。
11. 当前权值：此次随机生成的魔物共享的总权值。生成魔物的列表中，权值总和不得大于此值。
12. 总波数：此关卡中出现的大规模怪物的次数。
13. 大规模魔物数量增幅：当此次随机生成魔物为大规模魔物时，其值相当于**（8）**魔物最终最小出怪量 和 **（9）**魔物最终最大出怪量 同时增加的值。
14. 大规模魔物：当**（3）**关卡进度 每次达到 1 / **（12）**总波数 时，会触发大规模魔物事件。大规模魔物刷新时有以下特性：
15. 此次随机生成魔物中，**（11）**当前权值 将会提高到1.5倍。
16. 此次随机生成魔物时，**（8）**魔物最终最小出怪量 和 **（9）**魔物最终最大出怪量会同时增加一个固定值，值的大小为**（13）**大规模魔物数量增幅。
17. 此次随机生成的魔物数量不会被记录到**（2）**当前生成数量 中。
18. 当前波数：目前已经经历过的大规模怪物的次数。通常情况下，只有该值与**（12）**总波数 值相等时，关卡才会结束。
19. 魔物出现概率：是一个浮点数数组，其中包含着此关卡所有种类的魔物出现的概率。
20. 魔物权值：是一个整数数组，其中包含着此关卡所有种类的魔物的权值。

算法说明如下：**（1）**魔物出现总数 、**（4）**魔物生成间隔 、**（5）**魔物真实最小出怪量 、**（6）**魔物真实最大出怪量 、**（7）**魔物出怪修正值 、**（12）**总波数、**（13）**大规模魔物数量增幅、**（16）**魔物出现概率、**（17）**魔物权值 均在关卡中已事先给定。算法步骤为：

1. 计算此次生成魔物事件中，**（8）**魔物最终最小出怪量 和**（9）**魔物最终最大出怪量 的值。使用的公式为：
2. 计算此次生成事件中的**（11）**当前权值 。 每个关卡的计算方式可以不同，但一般为关于**（2）**当前生成数量 的单值线性函数。
3. 判断此次生成事件是否为大规模生成。当关卡进度大于**（15）**当前波数 /**（12）**总波数 时，此项为真，并执行对**（11）**当前权值 、**（8）**魔物最终最小出怪量 和**（9）**魔物最终最大出怪量的修改。
4. 随机生成**（10）**尝试生成数量。如果以下条件不能满足，则需要重新生成：
5. 根据**（10）**尝试生成数量和**（11）**当前权值 计算得到一个装载着魔物数据信息的数组，并将此数组返回给生成程序。计算的算法摘要：
   1. 获取**（17）**魔物权值中的最小值min和次小值less。同时记录最小值min和次小值less对应的魔物数据。
   2. 根据**（16）**魔物出现概率，随机生成一个魔物添加到列表，同时从当前权值中扣除此魔物的权值大小。
   3. 重复5.2操作，直至列表中魔物数据信息的数量等于**（10）**尝试生成数量，**或者**当前权值小于次小值less。
   4. 若是列表中魔物数据信息的数量等于**（10）**尝试生成数量，则返回列表信息；若当前权值小于次小值less，则继续进行以下操作：
   5. 直至列表中魔物数据信息的数量等于**（10）**尝试生成数量 ，**或者**当前权值小于最小值min 之前，将最小值min所对应的魔物数据添加到列表。结束时返回列表。
6. 得到魔物数据信息数组后，开始准备生成。若此次生成并非大规模生成，当场上魔物数量 < **（9）**魔物最终最大出怪量 时，进行生成；若此次生成为大规模生成，则等待到场上魔物数量为0时再开始生成。
7. 如果此次生成并非大规模生成，当列表内所有魔物生成完成后，**（2）**当前生成数量 增加，增加值为返回的魔物数据信息数组的长度。随后，程序睡眠一段时间，睡眠时间长度为**（4）**魔物出现间隔。

### 3.6.2 游戏控制模块：游戏物体控制算法

由于游戏中物体对象的重复性，对于游戏对象的重复生成和销毁会损失一定的程序运行速度。在这些游戏物体的生命周期结束后，并非将他们销毁而是将他们储存起来，能够节省一定的运算内存开销。对于游戏中随处可见的飞行类子弹、可以点击的能量、和元素反应的创造物草原核、结晶碎片等多种游戏对象，都采用了对象池的算法。

对象池简单介绍：主要内存占用为一个以游戏物体原型为键（Key），以复制出来的游戏物体列表为值（Value）的哈希表。对象池维护以下接口：

1. 获取一个池子中的游戏物体：利用游戏物体原型（Key）来访问哈希表中复制出来的游戏物体列表。如果列表中有正在休眠、可用的游戏物体，会将其激活，从列表中移除，并返回给外部；如果列表中没有可用的游戏物体，会使用游戏物体原型来克隆一个游戏物体返回出去。
2. 放置一个游戏物体：利用游戏物体原型（Key）来访问哈希表中复制出来的游戏物体列表。并将此游戏物体失活，纳入可用列表之中，下次外部再次需要此类游戏物体时，便可以通过1方法来获取一个。

但仅使用对象池仍无法实现对场景中所有可销毁物体的管理。由于游戏的特殊性，游戏可以暂停、重开、和中断并切换关卡，此时场景上的游戏物体应该被删除。因此游戏需要一个中枢控制模块来控制所有藉由对象池生成的临时游戏对象。

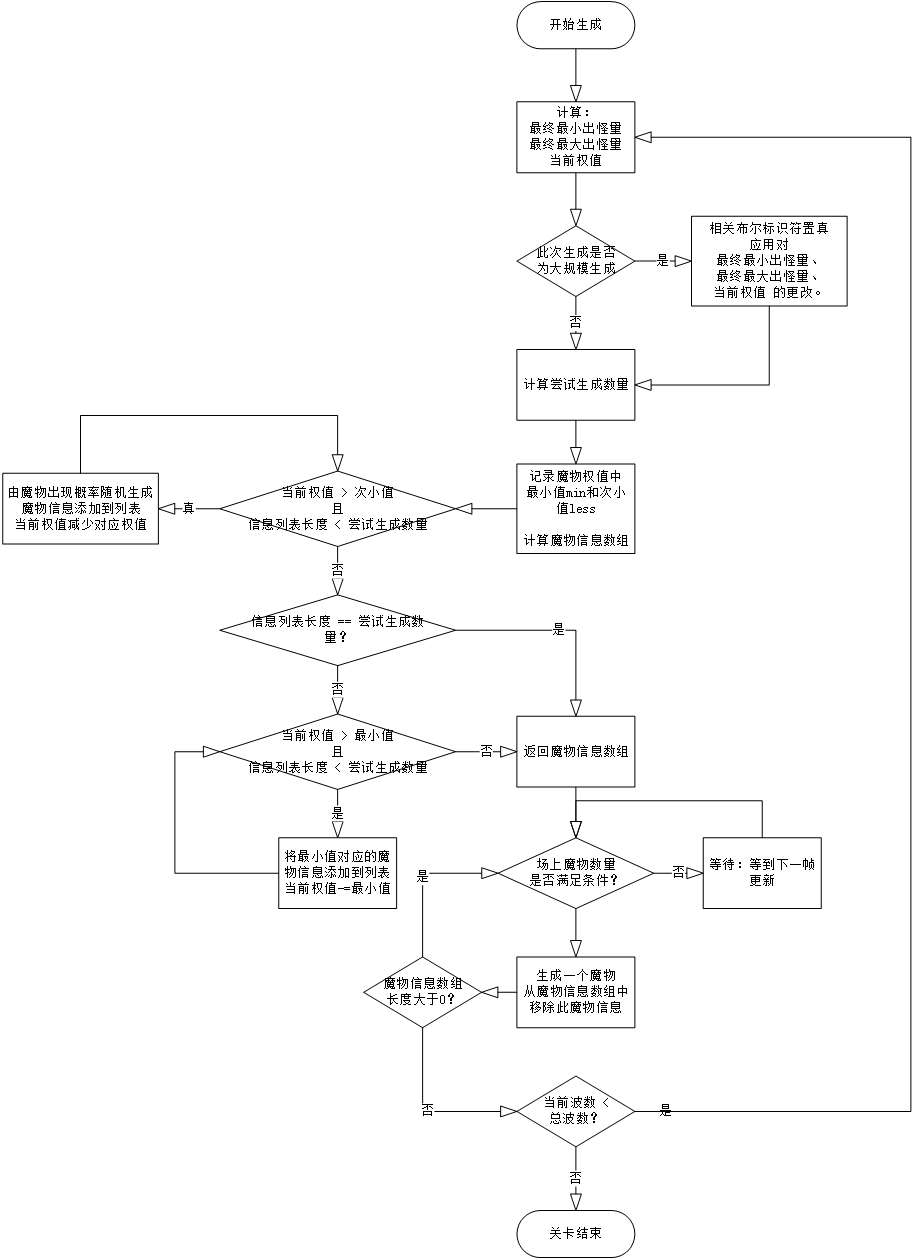
中枢控制模块维护一个以字符串string为键（Key），以集合Union为值（Value）的哈希表。集合Union包含以下内容：对象池指针、已生成对象列表。其中由于对象池本身是一个哈希表，因此已生成对象列表是一个以游戏物体为键（Key），以复制出来的游戏物体列表为值（Value）的哈希表。利用这样的数据结构，外部可以通过中枢控制模块维护的方法来将一个对象池添加到控制中去，方便游戏控制模块去遍历管理正在激活状态的游戏物体。

### 3.6.3 游戏逻辑模块.魔物逻辑模块：魔物状态控制算法

魔物状态算法主要维护了一个简单的有限状态机。对于此状态机，由于unity引擎自带的帧更新函数，并没有为状态机本身添加更新函数，而只增加了进入状态和离开状态的监听。此状态机仅仅维护了一个有向图，当某事件发生时，会判定此事件对应的边是否满足移动条件（当前所处状态是否是边的起始状态），如果可移动，则状态会改变。因此此控制算法主要控制了事件发生的判定。

## 3.7 程序逻辑

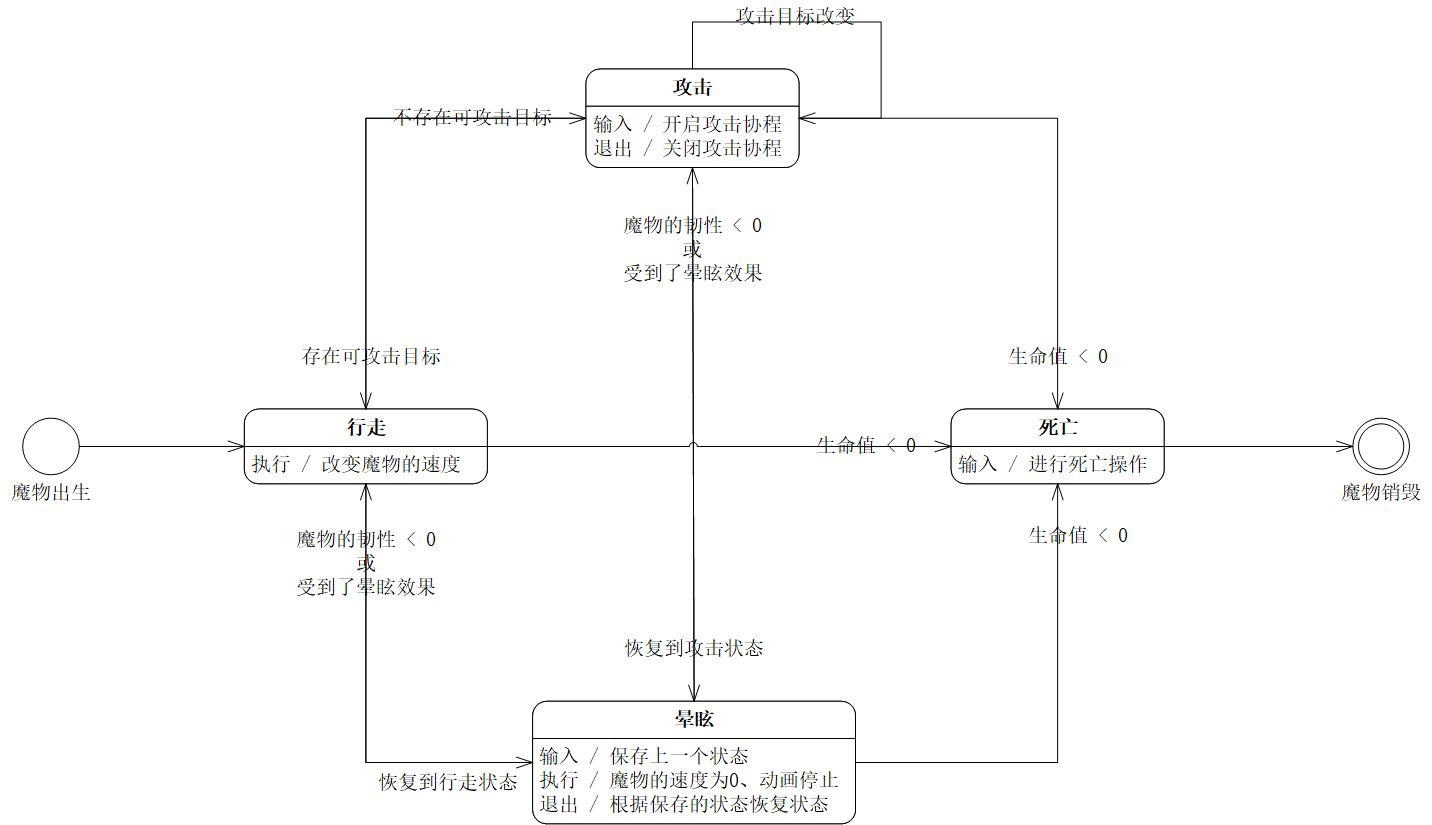
### 3.7.1 游戏逻辑模块.刷新模块：魔物刷新算法



### 3.7.2 游戏控制模块：游戏物体控制算法

## 此控制算法主要借助于中枢控制模块的遍历算法，将 正在激活中的游戏物体销毁。

### 3.7.3 游戏逻辑模块.魔物逻辑模块：魔物状态控制算法



## 3.8 接口

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **构件接口列表** | | | | |
| **模块名称** | **接口编号** | **接口名** | **接口类型** | **说明** |
| 输入解析模块 | 1 | OnUIElements  Triggered | 外部 | 当Unity的UI元素被触发时，调用此函数来解析。 |
| 植物数据模块 | 1 | AddPlants | 内部 | 添加植物 |
| 2 | RemovePlants | 内部 | 删除植物 |
| 3 | SearchPlant | 内部 | 查找植物 |
| 飞行物  数据模块 | 1 | AddFlyer | 内部 | 添加飞行物 |
| 2 | RemoveFlyer | 内部 | 删除飞行物 |
| 魔物数据模块 | 1 | AddMonster | 内部 | 添加魔物 |
| 2 | RemoveMonster | 内部 | 删除魔物 |
| 3 | SearchMonster | 内部 | 查找魔物 |
| 植物选择模块 | 1 | PlantOfIndex | 内部 | 卡槽中编号为i的植物 |
| 2 | IsReadyOfIndex | 内部 | 卡槽中编号为i的植物是否可用 |
| 能量模块 | 1 | AddEnergy | 内部 | 增加能量 |
| 2 | RemoveEnergy | 内部 | 删除能量 |
| 3 | TryRemoveEnergy | 内部 | 尝试移除 |
| 4 | Energy | 内部 | 能量 |
| 植物逻辑模块 | 1 | Action | 内部 | 植物触发逻辑 |
| 魔物逻辑模块 | 1 | Action | 内部 | 魔物触发逻辑 |
| 飞行物  逻辑模块 | 1 | Action | 内部 | 飞行物触发逻辑 |
| 死亡逻辑模块 | 1 | Update | 内部 | 触发死亡逻辑 |
| 刷新模块 | 1 | Update | 内部 | 刷新游戏 |
| 关卡IO模块 | 1 | WriteLevel | 内部 | 关卡文件写入 |
| 2 | ReadLevel | 内部 | 关卡文件读取 |
| 角色IO模块 | 1 | WriteCharacter  Data | 内部 | 角色文件写入 |
| 2 | ReadCharacter  Data | 内部 | 角色文件读取 |
| 关卡数据模块 | 1 | TransformPosition | 内部 | 变换坐标 |
| 2 | Row | 内部 | 行数 |
| 3 | Col | 内部 | 列数 |
| 4 | MonsterQueue | 内部 | 怪物出现队列 |
| 角色数据模块 | 1 | Health | 内部 | 角色生命值 |
| 2 | AtkPower | 内部 | 角色攻击力 |
| 元素数据模块 | 1 | GetElement | 内部 | 获取元素 |
| 2 | GetElement  Reaction | 内部 | 获取元素反应 |

## 3.9测试要点

采用黑盒测试方法，将植物放置，能量收集，植物逻辑，魔物逻辑，飞行物逻辑，元素反应正确触发产生效果作为重点测试要点