



# Terraria 电路文档

作者: putianyi888

组织: Terraria 电路爱好者交流群 (231355279)

时间: September 30, 2019

版本: 2.8.1



$\LaTeX$  模板: *ElegantBook*

# 目 录

<b>1</b>	<b>从零开始</b>	<b>1</b>
1.1	前言	1
1.2	一些基本概念与机制	1
<b>2</b>	<b>电路基础</b>	<b>6</b>
2.1	电源/电线/用电器	6
2.2	电源详解	15
2.3	用电器详解	18
2.4	逻辑门灯/逻辑门	23
2.5	故障逻辑门	27
<b>3</b>	<b>逻辑结算</b>	<b>36</b>
3.1	用 MechScope 模组研究电路结算过程	36
3.2	逻辑延迟器	42
3.3	普通逻辑门的逻辑同步	43
3.4	爆门	43
3.5	状态表示与激活表示	43
3.6	密码门	46
3.7	随机分两组	46
3.8	随机分三组	49
3.9	二进制加减法计算器	51
3.10	更细致的结算顺序	53
3.11	思考题	56
<b>4</b>	<b>数字电路</b>	<b>57</b>
4.1	布尔代数	57
4.2	组合逻辑与时序逻辑	62
4.3	计数系统	62
4.4	逻辑电路的运算速度	64
4.5	算术电路	64
4.6	存储电路	67
4.7	分段显示器化简理论	68
4.8	处理器结构	68

<b>5 电路文档</b>	<b>70</b>
5.1 传感器	75
5.2 递次电路	75
5.3 降频电路	75
5.4 数字显示	75
5.5 随机电路	75
5.6 操纵板	75
5.7 像素盒显示器	77
5.8 矩阵显示器	77
5.9 算术电路	77
5.10 存储器	77
<b>6 更多电路专题</b>	<b>79</b>
6.1 驱动与延时器	79
6.2 传感器	81
6.3 网线	83
6.4 传送阵	85
6.5 显示器	86
6.6 自动化	94
6.7 思考题	94
<b>7 农场</b>	<b>95</b>
7.1 水晶碎块农场	95
7.2 全自动树场	97
7.3 生命果/世花农场	97
7.4 液体反应池	97
7.5 松露虫农场	97
7.6 采沙场	97
7.7 天梯神教	97
7.8 南瓜神教	97
7.9 全自动月亮事件	97
7.10 DPS 纪录	97
7.11 速度纪录	97
7.12 Boss 速杀场地	97
<b>A 环境判定</b>	<b>98</b>
<b>B 刷怪机制</b>	<b>99</b>
B.1 刷怪率和刷怪量	101
B.2 刷怪点和刷怪面	103

B.3	刷怪类型	103
B.4	刷怪失败	105
B.5	刷怪种类	106
B.6	史莱姆雨刷怪	130
B.7	特殊判定	131
<b>C</b>	<b>NPC 无敌帧机制</b>	<b>132</b>
C.1	引入	132
C.2	说明	132
C.3	射弹造成的无敌帧	133
C.4	近战武器造成的无敌帧	136
C.5	零散内容	136
<b>D</b>	<b>液体</b>	<b>138</b>
D.1	全图液体刷新	138
D.2	将某格加入活跃液体数组	140
D.3	单格活跃液体刷新	140
D.4	熔岩反应判定	141
D.5	将某格从活跃液体数组中删除	142
<b>E</b>	<b>泰拉瑞亚中的常用变量与特殊集合</b>	<b>143</b>
E.1	常用变量	143
E.2	图格集合	143
E.3	背景墙集合	144
E.4	NPC 集合	145
<b>F</b>	<b>常用工具与网址</b>	<b>146</b>
F.1	Wiki	146
F.2	论坛	146
F.3	地图编辑器	146
F.4	模组	146
F.5	源码	146
<b>G</b>	<b>你应该知道的黑科技</b>	<b>147</b>
G.1	TNT 神教	147
G.2	传送器浮空	147
G.3	传送枪加速	147
<b>H</b>	<b>高质量电路作品视频</b>	<b>148</b>
H.1	视觉效果	148

H.2	解谜冒险	148
H.3	刷怪刷物品	149
<b>I</b>	<b>待实现的电路或装置</b>	<b>150</b>
I.1	电子钟	150
I.2	确定有限状态自动机	150
I.3	俄罗斯方块	150
I.4	贪吃蛇	150
I.5	魔方	150
I.6	Flappy Bird	151
I.7	2048	151
I.8	计算机	151
<b>J</b>	<b>冷却时间机制</b>	<b>152</b>
<b>K</b>	<b>机关射弹的生成和刷新机制</b>	<b>153</b>
K.1	匀速直线运动的射弹及其速度大小	153
K.2	抛射型射弹的运动机制	153
K.3	炮台发射的射弹的初速度方向	153
K.4	射弹碰撞箱及初始生成位置	154
K.5	实验测定抛射射弹轨迹	154
<b>L</b>	<b>网站代号</b>	<b>155</b>
<b>M</b>	<b>人名代号</b>	<b>156</b>



# 第 1 章 从零开始

---

## 1.1 前言

本书定位为“文档”，主要用于系统性收录电路理论，因此行文中会先讲理论后讲例子。如果读者觉得理论难以理解，不妨先看例子，再结合例子看理论。

每章后的思考题，有部分是经典电路的分析理解，有部分是因为我懒而没有去做的电路，还有一部分是纯理论推导。它们的共同点就是做不做都无所谓。与思考题相比，正文中用于举例、自成一节的电路是必须熟练掌握的。

因为泰拉瑞亚官方资料都是全英文的，并且几乎所有英文资料都没有翻译，所以在对泰拉瑞亚进行深入研究的时候请务必备好词典以及初中以上的英文水准。同时，一定的计算机或数学专业知识也会有帮助。如果你在词典的帮助下仍然看不懂英文，建议找人求助，而不是去使用机翻，机翻基本上没有一句话是准确的。<https://github.com/putianyi889/TMECbackup>整理了泰拉瑞亚英文官方论坛电路版块的部分帖子，你可以在那里求助翻译。

如果你对于纯文字的内容难以接受，也可以去观看视频教程，链接在附录中。视频相对于文字的缺点主要是时效性，因为视频不易更改，所以视频中的技术往往是已被淘汰的技术。我们仍建议在理解了视频教程的内容后以本书作为主参考。

本书中所有游戏名词首选为**Steam 平台**上最新版本**泰拉瑞亚**的中文，其次是**中文 wiki**。此外，对于想做大型装置的同学，**地图编辑器**与模组<sup>1</sup>是必不可少的，它们可以帮助你快速建造、备份。

关于游戏机制，如果你有编程基础，看反编译的**c# 源码**是最可靠的方法。否则请参考**Wiki**，对 Wiki 有疑问的话再求助可以看懂源码的人。

本书正文部分用于集中讨论电路，对于电路以外的信息会在附录中讨论。

在<https://github.com/putianyi889/TerrariaWiringTutorial>协助编写本书是唯一的**支持我们的方式**。你可以主动创作，也可以在**Issues**中领取任务。如果你不会使用 GitHub，可以看教程<https://zhuanlan.zhihu.com/p/34693871>。关注 (Watch) 本书的 GitHub 项目可以即时获取更新信息。

## 1.2 一些基本概念与机制

### 1.2.1 实体

实体指的是可以发生碰撞的物体<sup>2</sup>，包括但不限于**NPC**、**玩家**、**射弹**、**物品**、**图格**。

---

<sup>1</sup>**tModLoader**、**CheatSheet**、**HERO's mod**

<sup>2</sup>严格地讲，实体是编程术语，这里仅仅是在不影响游戏理解的前提下进行简化。

史莱姆对玩家造成接触伤害，就是史莱姆（NPC）与玩家的碰撞；玩家用弓射出木箭击中了史莱姆，就是史莱姆（NPC）与木箭（射弹）的碰撞；玩家被血肉墙激光击中，就是玩家与激光（射弹）的碰撞；玩家、掉落物、大多数 NPC、大多数射弹不能穿墙，是因为玩家、掉落物、NPC、射弹会和图格碰撞。

碰撞是通过碰撞箱判定的，例如史莱姆与玩家碰撞，是因为史莱姆的碰撞箱与玩家的碰撞箱有重叠。泰拉瑞亚中所有实体的碰撞箱均为矩形，有宽度和高度两个属性，它们可以在[源码](#)中查到。

1.2.2 硬上限与软上限

泰拉瑞亚中许多实体都有数量上限，而数量上限又分为硬上限和软上限。硬上限是游戏中的静态常量，例如 NPC 硬上限 200 等。软上限是游戏中的变量，例如活跃刷怪数量（一般在 5 到 15）等。

从程序角度来说，硬上限是由 C# 定长数组的长度决定的，如果尝试突破会导致数组越界。为避免在正常游戏过程中出现崩溃现象，游戏程序中在关键函数中都有越界检查，例如 NPC 达到上限时，游戏会拒绝生成新 NPC 以防止崩溃。软上限是开发者对游戏的平衡控制，例如玩家附近的活跃敌怪数量到达 15 则不会进行刷怪。

目前已知的硬上限见[表 1.1](#)。

表 1.1: 泰拉瑞亚中的硬上限

对象	上限
buff	22
傀儡影子	100
NPC	200
玩家	255
掉落物	400
射弹	1000
冷却机关	1000
宝箱	1000
活跃液体	5000
缓冲液体	10000

1.2.3 图像帧/物理帧

图像帧 (frame) 指泰拉瑞亚游戏过程中电脑显示屏更新的每帧画面，在游戏中按 F10 可以在游戏窗口左下角显示当前图像帧率。

物理帧 (tick) 指泰拉瑞亚中时间的最小单位，为 1/60 秒。泰拉瑞亚中所有碰撞/刷怪判定都是以物理帧为单位进行。由于本书是针对游戏机制的讨论，未经特殊说明的情况下将直接用“帧”表示物理帧。



关于图像帧与物理帧的详细对比，请参阅<https://www.bilibili.com/video/av22788696> 1 分 53 秒处。

### 1.2.4 坐标

坐标就是在世界中的位置。坐标并非深度计与罗盘所显示的那样。程序中的坐标是以世界左上角为 (0,0)，横坐标向右，纵坐标向下。

泰拉瑞亚中纵坐标分层一般分为太空、地表、地下、洞穴、地狱五层。而游戏机制中，只有两个阈值，一个是地表层与地下层交界处的纵坐标，称为 `worldSurface`；另一个是地下层与洞穴层交界处的纵坐标，称为 `rockLayer`。太空层高度是把地表之上总高度（即地表的纵坐标）乘上一个系数得到的；地狱层高度是把世界底端坐标（即世界总高度）减去一个常数得到的。这个系数一般是 0.35，常数一般是 200 格，但是在不同情况下也可能会有出入。

### 1.2.5 度量

泰拉瑞亚中的长度单位有：英里、格、英尺、像素。换算关系是 1 英尺 = 8 像素 = 1/2 格 = 1/5280 英里。

泰拉瑞亚中的时间单位有：帧、秒、分、天。换算关系是 1 天 = 24 分，1 分 = 60 秒，1 秒 = 60 帧。有的时候帧、秒、分也分别叫做游戏秒、游戏分、游戏时。

速度单位 = 长度单位/时间单位，主要有：英里/小时、格/秒、像素/帧。

程序内部的长度单位是像素，时间单位是帧，速度单位是像素/帧。

### 1.2.6 驱动

驱动 (engine) 指可以间歇性自动激活电路的装置。驱动按频率分类可分为低频驱动、高频驱动、满频驱动、超频驱动。

- 低频驱动指频率小于等于 1Hz 的驱动。这类驱动一般通过计时器降频得到。
- 高频驱动指频率大于 1Hz 且小于 60Hz 的驱动。这类驱动造法非常丰富，最可靠稳定的方法是利用满频驱动降频。
- 满频驱动指频率等于 60Hz 的驱动。主流的满频驱动有假人驱动和传送带驱动。之所以叫满频驱动，是因为驱动频率与物理帧率相同。更高的频率也可以通过满频驱动得到相同的效果。例如，120Hz 的驱动的输出效果和两个满频驱动同时输出的效果完全相同。
- 超频驱动指频率大于 60Hz 的驱动。此类驱动一般用多个满频驱动同时运行，或者利用加重压力板的超灵敏度。超频驱动可用于驱动计算装置。

### 1.2.7 半砖

当掉落物/非穿墙生物的碰撞箱与实体块重合时，程序会尝试将碰撞箱推离实体块。从 1.2 版本开始，大多数前景物块都有六种半砖形态，每种半砖推离碰撞箱的机制各不



相同。尽管半砖在电路中占有一席之地，由于其：本身不涉及到电路；应用不广泛<sup>3</sup>；目前没有严谨的机制；设计装置主要靠经验和尝试，本书中暂时不涉及半砖教学。

关于半砖有关的研究与教程，读者可以参考以下的链接，链接排序随机。

- 视频
  - 你会使用半砖吗?-Zerogravitas <https://www.bilibili.com/video/av22088325>
  - 物品半砖以及一些有趣应用!-Zerogravitas <https://www.bilibili.com/video/av22739847>
  - 鬼畜的石巨人! -Zerogravitas <https://www.bilibili.com/video/av22088547>
- 文章
  - 【实验】半砖性质探究 <https://tieba.baidu.com/p/3603529198>
  - HOIK! [Guide] - Rapid Player/NPC/Etc Transport Using Only Sloped Tiles. <https://forums.terraria.org/index.php?threads/hoik-guide-rapid-player-npc-etc-transport-using-only-sloped-tiles.1656/>
  - [Early Game] Anti-Monster Wall Defense (Using HOIK! and Stair Glitch) <https://forums.terraria.org/index.php?threads/early-game-anti-monster-wall-defense-using-hoik-and-stair-glitch.29917/>
  - Hoik tracks with pressure plates for mounts <https://forums.terraria.org/index.php?threads/hoik-tracks-with-pressure-plates-for-mounts.37113/>

### 1.2.8 射弹生成以及刷新机制

射弹 (projectile) 是泰拉瑞亚中的一大类实体。包括但不限于机关射出来的飞镖、火焰，抛出的悠悠球，棱镜射出的激光，扔出的沙滩球，挥舞的日耀链刃，甚至玩家死亡后弹跳的墓碑。

这小节内容主要针对所有射弹的生成及刷新过程，用于后续某些内容的引用，读者大可直接跳过本小节。

游戏使用一个长度为 1000 的列表存储射弹。射弹生成时，其信息会被存储在射弹列表的第一个空位。如果射弹列表没有空位，那么该射弹不会生成。

每个物理帧，游戏会执行一轮射弹刷新过程，即对列表从头到尾扫描，对每个射弹执行其刷新函数。每个射弹的刷新方式取决于射弹的 id。射弹刷新第一步是改变射弹的位置，即

$$\text{新位置} = \text{原位置} + \text{速度}。$$

对于非匀速直线运动的射弹，还需要对其速度进行计算。第二步就要进行碰撞判定，射弹如果与生物碰撞了，就可能要进行伤害计算；如果与前景物块碰撞了，就可能要销毁

<sup>3</sup>其大多数功能可以用传送机或传送带解决。

射弹：如果与青绿压力垫板碰撞了，就可能要插入电路结算。这最后一种情况就是这本书主要关心的内容。

机关射弹的刷新机制列举在附录中。



## 第 2 章 电路基础

### 2.1 电源/电线/用电器

在现实生活中，电路的三个基本组成部分为电源、导电回路和用电器，电源产生的电流经回路传导到用电器并驱动用电器。在泰拉瑞亚中，电路的三个基本组成部分为电源、电线和用电器，电源产生的电信号经电线传导到用电器，用电器响应该电信号。[wiki](#)上这样描述这个过程：电源激活——电源上的电线激活——电线下的用电器激活。在本书接下来的讨论中，我们采用 [wiki](#) 上的解释，统一使用激活 (activate) 来表达电路的活跃状态。

由于电线和图格处于不同图层，它们可以重叠。如果某电线与某图格重叠，我们说该图格在该电线下，该电线在该图格上。

电源指可激活其上电线的图格或该图格对应的物品，每个电源有其特有的激活条件。泰拉瑞亚中所有电源及其激活条件见表2.1，详细信息请参阅[节 2.2](#)及 [wiki](#)。

电源	物品贴图	激活条件
开关/控制杆		鼠标右击
灰/棕/蓝/丛林蜥蜴压力板		玩家踩踏
红/绿压力板		玩家/NPC/敌怪踩踏
黄压力板		NPC/敌怪踩踏
加重压力板		玩家踩上或离开
青绿压力垫板		射弹触碰
1/3/5 秒计时器		开启后每隔 1/3/5 秒激活
引爆器		玩家自上而下冲击或鼠标右击
宝石锁		对应的大宝石被嵌入或取出
受困宝箱		
逻辑感应器（昼/夜）		入昼/入夜
逻辑感应器（玩家）		玩家进或出蓝色方框
液体感应器		对应液体触碰
逻辑门		详见后文

表 2.1: 泰拉瑞亚中的电源

用电器指可被其上电线激活的图格或该图格对应的物品，每个用电器被激活时有其特有的响应方式。泰拉瑞亚中所有用电器及其响应方式见表2.2，详细信息请参阅[小节 5.2.1](#)及 [wiki](#)。

最简单的电路包含一个电源、一个用电器，以及连接电源与用电器的电线。在这个电路中，激活电源的瞬间，用电器会被自动激活。

广义的电源指一个起到电源作用的模块，例如假人驱动。广义的用电器指一个起到用电器作用的模块，例如显示屏。

有一部分用电器被激活的时候会在两个状态间切换，例如发光物品的亮灭切换，功能物品的开关切换。在涉及到逻辑的时候，可以将两个状态看作 1 和 0。

表 2.2: 泰拉瑞亚中的用电器

用电器	物品贴图	响应方式
火把		亮/灭切换
部分蜡烛		亮/灭切换
部分灯笼		亮/灭切换

用电器	物品贴图	响应方式
灯		亮/灭切换
篝火		亮/灭切换
烛台		亮/灭切换

用电器	物品贴图	响应方式
吊灯		亮/灭切换
晶莹宝石块		亮/灭切换
其他 单色 发光物		亮/灭切换
门		开/关切换
机关门/ 高门		开/关切换







用电器	物品贴图	响应方式
泵		把入水泵上的液体传送到出水泵
机关		发射射弹
炸药 地雷		爆炸并消失
炮台		根据激活点 改变方向/ 射击
烟花喷泉/ 烟花盒		产生烟花
烟花火箭		发射
泡泡机/ 呆萌气球机/ 派对中心		开/关切换
喷泉		开/关切换
八音盒		开/关切换



用电器	物品贴图	响应方式
部分雕像		生成物品/ 传送 NPC/ 开/关切换/ 生成敌怪/ 生成小动物
烟囱		三个状态 切换
传送机		交换两个 传送机上的 生物
天塔柱		开/关切换
广播盒		显示 文字信息
传送带		改变方向
制动器		切换前景 物块的 虚化状态
彩线灯泡		四色电线 各控制一个 灯泡的亮灭



用电器	物品贴图	响应方式
像素盒		从上/下激活时熄灭，同时从上/下和左/右激活时点亮
逻辑灯		详见后文
矿车轨道交叉点		在两种交叉方式中切换

2.1.1 让派对永不停止

在视频<https://www.bilibili.com/video/av21009075/>中，为了使所有 NPC 戴上派对帽，需要使地图保持在派对事件中。派对事件可通过派对中心激活。入夜时派对事件自动结束，派对中心也会关闭。想让地图保持在派对事件中，就需要在入夜时重新开启派对中心。一个简单的装置可以实现这个功能。如图 2.1 所示，用电线连接逻辑感应器（夜）与派对中心，组成一个完整的电路。这个电路中，逻辑感应器（夜）为电源，派对中心为用电器。入夜时，首先派对事件结束，派对中心关闭，然后逻辑感应器（夜）激活，派对中心激活，又重新打开。每天入夜时重复这个过程，从而使地图保持在派对事件中。需要注意的是该装置能够实现所需功能，依赖于入夜时事件结算先于感应器结算。如果感应器先结算，则该装置无法实现所需功能。



图 2.1

2.1.2 传送带驱动

传送带驱动是结构最简单、占地最小的满频驱动，其电路图见图 2.2。在这个电路中，加重压力板是电源，传送带是用电器。要触发此驱动，只需要玩家站在传送带上。传送带使玩家向右移动，玩家进入加重压力板所在格时激活加重压力板，进而激活传送带，传送带变为逆时针，玩家向左移动，移出加重压力板时又激活加重压力板，传送带变为顺时针，玩家向右移动，如此反复。红线上的信号即为驱动信号。需要注意的是，玩家进入压力板的“瞬间”即向左移动，移出压力板的“瞬间”即向右移动，所以该驱动的频率等于电脑判断玩家与压力板是否重叠的频率，即帧率。





图 2.2: 传送带驱动。图中传送带为顺时针。

### 2.1.3 自动门

很多人刚开始玩泰拉瑞亚的时候，盖房子都会造门。普通的门是非常不方便的，不光出入都需要操作，门的两边不能有障碍物，而且有些时候并不能阻止敌怪进入。解救了机械师后，自然就会把普通的门改成装有制动器的物块（图 2.3），这样的门两边可以有障碍物，而且血月也不会被开门。为了节省出入门的额外操作，就发明了自动门，它可以在玩家出入门时自动打开，出入门后自动关闭。

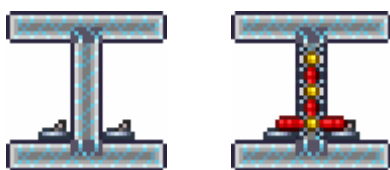


图 2.3: 使用制动器做的门，开关可以开关门。

使用两个压力板就可以完成这个功能。如图 2.4 所示，当玩家从左边走向门时，会踩到门左边的压力板，压力板激活制动器，把物块虚化。当玩家通过门时，踩到门右边的压力板，压力板激活制动器，把物块实化。玩家从右边向左走同理。



图 2.4

善于思考的人很快就会发现这个门的漏洞，那就是如果人走到门口，又回去，门就会保持开启状态。为了解决这个问题，需要把普通压力板改成加重压力板（图 2.5）。



图 2.5

然而改成了加重压力板后并非万事大吉。如果进入多人游戏，两个人站在门的两侧，同时试图到对方一边，那么制动器被激活两次，物块仍实化，除非一个人让步。这是非常不方便的。使用逻辑感应器（玩家）可以解决这个问题（图 2.6）。当已经有一个玩家站在感应器的蓝框内时，其他玩家进出不会改变感应器状态，感应器自然就不会激活。因此，当感应器的蓝框内有玩家时门开启，否则门关闭。

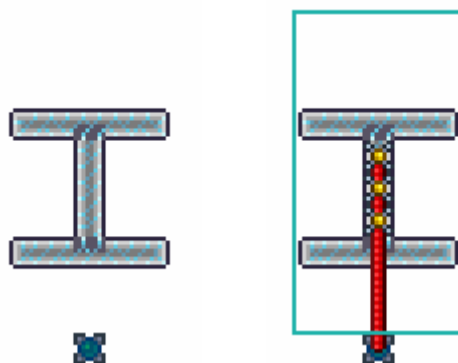


图 2.6

### 2.1.4 刷液体机

刷水机曾经是钓鱼党的必备，不过有了无底水桶后就彻底成为了电路党的专利。目前来说刷液体机主要用于会消耗液体的装置，例如 2D 打印机<sup>1</sup>、打开地图瞬间触发电路的模块、三液体混合反应池。另外也有人利用刷液体做一些考验电脑配置的事，比如水淹世界、水淹地狱等。

刷液体的基本原理就是泰拉瑞亚中液体的“玄学”流动机制。尽管 Appendix D 中介绍了液体的流动机制，但是由于游戏环境下涉及到的液体格数过多，模型过于复杂，目前尚无一个完善的理论来简单预测液体的行为。一般来说，分流会导致液体增加，而在平面上摊开会导致液体减少。刷液体机大多数都是利用前景物块分流液体。由于液体的机制尚不明确，刷液体机的分流构造也都是凭经验。这里介绍刷液体机侧重点在于激活电路的方式，因此分流结构从简。

首先我们用物块搭一个有分流装置的小池子，并在底部放上入水泵，顶部放上出水泵，设置一个 1 秒计时器（图 2.7(a)）。往池里倒上一定量的水，然后用电线连接入水泵、出水泵和 1 秒计时器（图 2.7(b)），右键打开 1 秒计时器，那么每隔一秒，入水泵上的水被传送到出水泵，刷水机就开始工作了。可以虚化一格池壁来取水。由于不同液体流速不同，刷熔岩建议使用 3 秒计时器，刷蜂蜜建议使用 5 秒计时器。

这种构造的刷液体机有一个小缺点，那就是需要手动打开计时器，因为计时器在退出地图时会自动关闭。另一方面，水充满池子的时候，水泵无效，但计时器仍在运行，这对于强迫症来说是一个打击。有一个方法可以解决这个问题，那就是在水泵边上设置液体感应器（图 2.7(c)，图 2.7(d)）。如果水位够高，液体感应器保持为亮，水泵不工作。如果取水使得水位降低低于液体感应器，那么液体感应器熄灭，激活水泵，水泵抽水，抽出的水经过液体感应器时点亮液体感应器，液体感应器又激活水泵，直到水位达到液体感应器时液体感应器才因保持为亮而不激活电路。这个装置不仅做到了智能刷水，而且液体感应器这个驱动频率可以完美适配液体流速。

<sup>1</sup>视频链接[https://v.youku.com/v\\_show/id\\_XMjUwNDY5MzM4OA](https://v.youku.com/v_show/id_XMjUwNDY5MzM4OA) 11 分  
地图下载<http://pan.baidu.com/s/1slbFpnR>

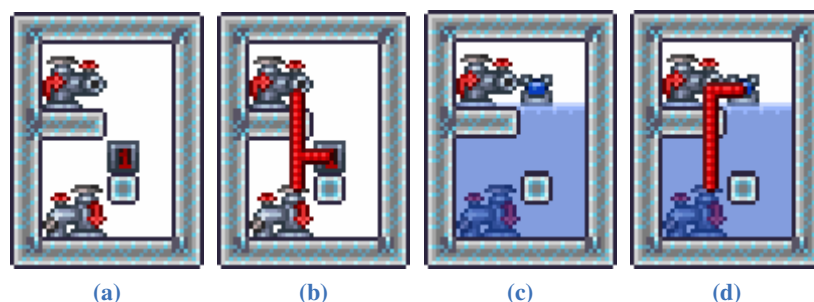


图 2.7

### 2.1.5 二进制数显

数字显示屏是一个装逼利器。事实上是，很多泰拉瑞亚视频中的数字显示屏都是整个电路中最简单的一个模块。二进制数显是最简单的一种数显。

首先用火把摆出数字“0”和“1”（图 2.8(a)）。我们要让火把可以在“0”和“1”之间切换，只需要用电线连接“0”比“1”多出来的火把（图 2.8(b)）。右击开关，这些火把就会在亮灭之间切换，从而数显在“0”“1”之间切换。

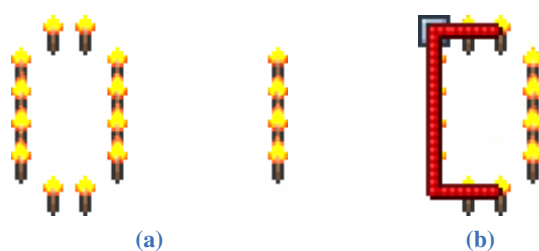


图 2.8

## 2.2 电源详解

### 2.2.1 开关/控制杆

开关和控制杆的激活条件是鼠标右击。同其他交互类物品/NPC 相同，只有当开关/控制杆在可触及范围内时才可以右击。

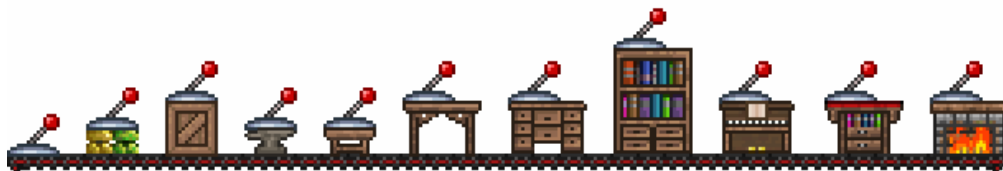
与控制杆相比，由于开关体积更小，在电路密集时一般都使用开关。另一方面，小的体积带来的缺点是开关不容易发现，而且容易点错。

开关可以放置在木梁侧面与前景物块侧面而控制杆不行；控制杆可以放置在平坦表面上而开关不行（图 2.9）。需要注意的是，只有当砧上有足够面积的背景墙时，控制杆才可以放置在砧上，放置在砧上后敲掉背景墙也不会掉落，这可能是判定的 bug。

### 2.2.2 压力板

压力板分为普通压力板、加重压力板、青绿压力垫板。





**图 2.9:** 放置在平坦表面上的控制杆

普通压力板包括由玩家触发的灰/棕/蓝/丛林蜥蜴压力板、由敌怪触发的黄压力板和由玩家或敌怪触发的红/绿压力板。加重压力板是误翻译，正确翻译应为“重力压力板”。四种颜色的加重压力板功能完全一样。

普通压力板有自身的碰撞箱，碰撞箱大小是压力板弹起状态时贴图的边框大小。普通压力板激活的判定以每个碰撞箱为准，即每帧判定碰撞箱是否从侧面进入压力板，或从上面掉落到压力板上。因为是以碰撞箱为准，所以一个碰撞箱在一帧内触发两个普通压力板，只会激活一次，而不同碰撞箱在一帧内触发同一个普通压力板，每个碰撞箱都会激活一次（图 2.10）。注意到“进入”或“掉落”都是过程，所以直接传送到普通压力板上不会触发压力板。同时，“掉落”要求人物有一个悬空的过程，判断悬空可以通过人物动作（悬空时有跳跃动作）或者翅膀（装备了翅膀的人物悬空时翅膀会打开）。所以从 1 格高的物块上直接走下来不算掉落，同时碰撞箱也不是从侧面进入，所以不会触发普通压力板（图 2.11）。



图 2.10: ((a))同时踩踏两个红压力板，只有左边的火把响应；((b))虚化两个 NPC 脚下的方块，两个 NPC 同时掉落到红压力板上，火把仍亮。



图 2.11: ((a))直接从一格高度走下，翅膀不打开，无跳跃动作，压力板不会被触发；((b))在平台上按“下”方向键，翅膀打开，有跳跃动作，压力板会被触发。

加重压力板没有碰撞箱，判定以压力板本身为准，即每帧或者每次传送后判定是否有人物在压力板所在格内，直接传送到加重压力板上可以触发加重压力板。为避免冲突，



加重压力板的激活信号及以该信号触发的逻辑结算产生的激活信号不会触发传送机。

青绿压力垫板的碰撞箱为  $16*10$  或  $10*16$ ，根据其朝向而定。射弹生成后，每次更新位置都会激活碰撞到的青绿压力垫板，这里碰撞的定义是：上次更新时碰撞箱不相交，但是这次更新时相交。如果射弹同时与多个青绿压力垫板碰撞，那么只激活优先级最高的那个（不同行的青绿压力垫板，上面的优先级高；同一行的青绿压力垫板，左边的优先级高）；多个射弹同时碰撞同一个青绿压力垫板，每个射弹都会激活一次（图 2.12）。

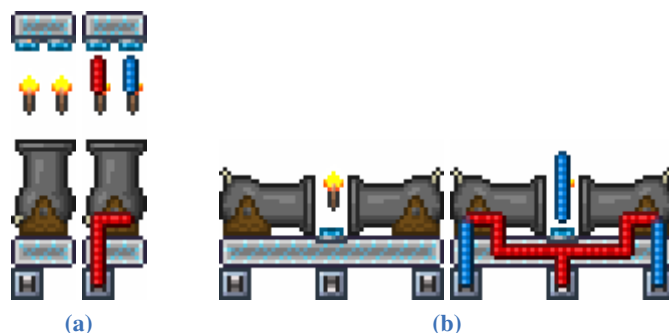


图 2.12: ((a))大炮发射，只有左边的火把响应；((b))激活左边或右边的开关火把都响应，激活中间的开关火把不响应（实际上响应了两次）。

普通压力板和加重压力板可以放在平台和锭上而青绿压力垫板不能；青绿压力垫板可以放在前景物块侧面和下方而普通压力板和加重压力板不能。

压力板轨道是带有压力板的轨道，它只能由矿车触发。

### 2.2.3 引爆器

引爆器是一个比较特殊的物品，其有压下和弹起两种状态。鼠标右击或人物以至少 3 像素/帧的垂直速度经过引爆器上两格时会导致引爆器压下并作为电源激活，随后经过 1 秒，引爆器自动弹起。可能出于判定原因，当人物以一定速度（既不快也不慢）落在引爆器旁边的支撑物边缘时引爆器也会压下（图 2.13）。引爆器处于压下状态时鼠标右击或人物踩踏均无效（贤者模式）。



图 2.13: ((a))站在平台边缘原地跳跃，当跳跃高度在某个范围内时，落下会踩下引爆器；((b))骑乘坐骑也有这种现象。

引爆器同样也是用电器，被激活时会在压下和弹起之间切换。如果引爆器被激活压下，那么不会作为电源激活，也不会自动弹起，直到再次被激活弹起。

引爆器可以放在几乎所有平坦表面上。与控制杆不同的是，引爆器不是悬挂家具，所以不能放在砧上。

### 2.2.4 受困宝箱

受困宝箱是误翻译，正确翻译应该是“机关宝箱”或“陷阱宝箱”。受困宝箱是电源，当鼠标右击时激活并播放宝箱开启关闭的动画。除了右击效果不同以外，受困宝箱与对应的普通宝箱外观、放置方式完全相同，这使得受困宝箱可以用来做陷阱的触发器。

### 2.2.5 感应器

泰拉瑞亚中共有 3 种逻辑感应器和 4 种液体感应器。逻辑感应器（昼）在白天点亮，夜晚熄灭；逻辑感应器（夜）在白天熄灭，夜晚点亮；逻辑感应器（玩家）放置时对应一个电路层的蓝色方框，该方框宽略大于 5 格，高略大于 10 格，当框内有玩家时感应器点亮，当框内无玩家时感应器熄灭。液体感应器在所在格中有对应液体时点亮，无对应液体时熄灭。

逻辑感应器（昼）和逻辑感应器（夜）在由灭变亮时激活，其他感应器均在亮灭切换时激活。

逻辑感应器（玩家）可以看作感应范围更大，但是对传送不敏感，仅在每帧进行判断的加重压力板。逻辑感应器（玩家）的激活信号及以该信号触发的逻辑结算产生的激活信号不会触发传送机。

所有感应器均可随意放置，无需支撑块或背景墙。

使用地图编辑器或 Mod 复制粘贴的感应器无法工作，需拆除后手动放置。因此设计电路时应尽量避免大规模使用感应器。

### 2.2.6 其他电源

剩余的电源在 wiki 之外的信息相当少，因此不专门介绍。它们是：计时器、宝石锁。

## 2.3 用电器详解

### 2.3.1 部分光源

这里说的光源不是指有效光源，而是指所有能发光的物体。所有可由电路控制的光源如下列举。

- 火把：大小 1\*1，放置在平台上、前景物块上方和两侧、背景墙上。是有效光源。
- 蜡烛：大小 1\*1，放置在除砧以外的平坦表面上，可以放置在砧上但会立刻掉落。水蜡烛和和平蜡烛无电路功能。是有效光源。
- 灯笼：大小 1\*2，悬挂在前景物块下。萤火虫瓶和荧光虫瓶无电路功能。星星瓶和红心灯笼熄灭时不提供 buff。是有效光源。
- 灯：大小 1\*3，放置在平台上、砧上、前景物块上。是有效光源。

- 篝火：大小 3\*2，放置在平台上、锭上、前景物块上。熄灭时不提供 buff。是有效光源。
- 烛台：大小 2\*2，放置在除砧以外的平坦表面上、前景物块上。是有效光源。
- 吊灯：大小 3\*3，悬挂在前景物块下。是有效光源。
- 晶莹宝石块：属于前景物块。不是有效光源。
- 中式灯笼：大小 2\*2，悬挂在前景物块下。是有效光源。
- 灯柱：大小 1\*6，放置在平台上、锭上、前景物块上。不是有效光源。
- 迪斯科灯：大小 2\*2，悬挂在前景物块下。不是有效光源。
- 圣诞灯：大小 1\*1，放置在前景物块四周。是有效光源。
- 壁炉：大小 3\*2，放置在平台上、锭上、前景物块上。是有效光源。

这里需要强调一下常用的两种显示光源：宝石块和火把。一般情况下宝石块显示效果更好，并且其亮灭会显示在小地图中。然而由于火把激活时只是简单改变状态，而每个宝石块激活时还需要更新该块及周围 8 块的贴图，每次更新贴图时都需要进行大量的判断，这导致大规模使用宝石块的电路与使用火把的电路相比非常卡。

### 2.3.2 门、机关门、高门

上锁的丛林蜥蜴门无电路功能（废话）。门放置在上下两个前景物块之间。图格、生物出现在门一侧的开启范围内（1\*3）时门不会向这一侧开启；出现在门两侧开启范围内时门无法开启。当门可以向两侧开启时，如果右键开门，那么门向玩家面对方向开启；如果电路开门，那么门似乎是随机向两侧开启。

与门相比，机关门的开启方向是上下。当机关门可以向两侧开启时，使用右键开门，根据玩家与门的相对位置确定开门方向：玩家在相对高处时门向下开启；玩家在相对低处时门向上开启。使用电路开门，固定向下开启。

高门没有方向性，开门也不受阻挡。

### 2.3.3 泵

用一根电线连接一个入水泵和一个出水泵，则电线激活时，入水泵上的液体会尽可能多的转移到出水泵。

为了了解一些奇怪情况下的液体转移结算，这里介绍水泵的内部运行机制。

游戏中用两个长度 19 的列表分别存储入水泵和出水泵的坐标。在这个列表中，每个泵都是单个图格。游戏中的泵包含四个图格，当该泵被激活时，四个图格被依次加入列表中，顺序是左下-右下-左上-右上。加到列表满时则不继续加入。

每根电线结算完成后进行水泵结算，从前往后扫描入水泵列表（没有液体的入水泵除外），对于每个入水泵，从前往后扫描出水泵列表（满液体的出水泵除外），并对入水泵和出水泵中的液体进行转移。

单格入水泵和单格出水泵之间的液体转移，首先要遵循液体一致的原则，即转移液体不会导致不同液体出现在出水泵上。其次，转移的量为入水泵上的液体总量和出水泵上空余液体量的最小值（每格中的液体量为 0 到 255）。

### 2.3.4 机关

这里说的机关指激活会对玩家造成伤害的用电器。

超级飞镖机关、尖球机关、烈焰机关、长矛机关都生成在丛林蜥蜴神庙内。飞镖机关生成在地下其他位置。这 5 种机关属于前景物块，锤击可以改变射击方向；被激活会生成射弹，可触发青绿压力垫板。

飞镖机关、超级飞镖机关、尖球机关和烈焰机关的射弹在机关前方第 2 格生成，因此紧贴这 4 种机关前方的一个前景物块不会阻挡射弹。紧贴长矛机关前方的前景物块会阻挡长矛。

飞镖在真空中的速度是每秒 45 格，生存时间 60 秒。飞镖机关、超级飞镖机关和烈焰机关的冷却时间是 200 帧。

长矛机关射程 19.5 格<sup>2</sup>，冷却时间 90 帧。

烈焰机关射程 20 格，冷却时间 200 帧。尽管烈焰机关的特效较宽，其射弹仍只在机关正前方生成。

尖球机关冷却时间 300 帧，有生成限制。其限制规则较复杂，请参阅 wiki。

因为翻译原因，喷泉（机关）在中文 wiki 中无法查到，因为与喷泉（装饰）冲突<sup>3</sup>。请在英文 wiki 中查阅 Geyser 词条。

喷泉（机关）可放置在前景物块上或下，其朝向也根据放置位置分为朝上和朝下。冷却时间 200 帧。不会触发青绿压力垫板。射程 20 格，射程从喷射方向 29 格内遇到的第一个 2\*4 开放空间（没有前景物块和液体）开始，其后会被障碍物阻挡。

炸药被激活时爆炸，爆炸半径 10 格。炸药是一次性的，所以在电路中应用有限。使用地图编辑器或 Mod 可以将炸药放在宝箱下，由于宝箱下的物块无敌，炸药可以被无限引爆。

地雷被激活时爆炸。与炸药的区别是地雷不破坏图格，伤害也更低。另一方面，地雷被玩家或 NPC 踩踏时也会爆炸，这使得引爆地雷可以不用电线，因此可以通过刷漆使地雷完全不可见。

### 2.3.5 炮台

炮台包括大炮、兔兔炮、彩纸炮、传送枪站和雪球发射器。

炮台大小为 4\*3，但是可以放置在两格宽的前景物块、平台或锭上。

鼠标右击炮台左/右部分可转向，拿着对应炮弹左击可发射。使用电线激活时，激活炮台的不同位置有不同效果（图 2.14）。使用电线激活发射的炮弹不造成伤害。

炮台生成射弹的位置横坐标在中间两列的交界，纵坐标在下面一格与中间一格的交界。对于传送枪台，生成位置还要向下移 5 像素；如果传送枪台朝向垂直向上，生成位置还要向右移 5 像素。

炮台生成射弹的初速度大小，传送枪站为 93 像素/帧<sup>4</sup>，其他为 14 像素/帧。初速度

<sup>2</sup>由 @dcfht 验证。

<sup>3</sup>喷泉 (Geyser) 是自然景观，喷泉 (Fountain) 是人造景观，官中并未区分翻译。

<sup>4</sup>具体为每次前进 3 像素，每帧前进 31 次

方向只与炮台朝向有关，除了  $0$ 、 $\pi/2$ 、 $\pi/4$  的特殊角以外，非特殊角分别为  $\arctan(1/3)$  和  $\pi/2 - \arctan(1/3)$  而不是  $\pi/8$  和  $3\pi/8$ 。

大炮和兔兔炮的炮弹射出后，前 17 帧速度不变，从第 18 帧开始，每帧纵向速度增加 0.28 像素/帧（最大 16 像素/帧），横向速度乘 0.99。传送枪台的射弹匀速直线运动，并会按顺序激活路径上的所有青绿压力垫板。彩纸炮发射的射弹匀速直线运行，生存期为 2 帧，生存期结束后爆炸，生成特效。在这 2 帧内，射弹只能前进 28 像素，无法离开炮台  $4*3$  的大小，也就无法触发青绿压力垫板。



图 2.14: 红线右转，蓝线左转，绿线发射，黄线改变射弹颜色。

传送枪站的鼠标操作与其他炮台不同。鼠标右击传送枪站不同位置效果与电线激活对应位置相同。

除彩纸炮以外的炮台发射的射弹可以触发青绿压力垫板。冷却时间 30 帧。

雪球发射器较特殊，其特性与以上描述几乎完全不同。雪球发射器大小  $3*3$ ，只能放置在三格宽的前景物块、平台或锭上。雪球发射器不可手动转向，背包中有雪球时右击可发射雪球，可连发。发射出的雪球可触发青绿压力垫板。使用电路激活其左三格之一会使其朝向左，激活其右三格之一会使其朝向右，激活中间三格之一会发射不造成伤害的雪球。雪球发射器朝向只有两种：水平向左和水平向右。发射点在雪球发射器中心向左 12 像素（如果朝向左）或向右 12 像素（如果朝向右）；发射速度在  $[12:0.01:16.49]$  中随机；发射方向随机（见图 2.15）。雪球发出后，前 19 帧速度不变，从第 20 帧开始，每帧纵向速度增加 0.3（最大 16），横向速度乘 0.98。冷却时间 10 帧。

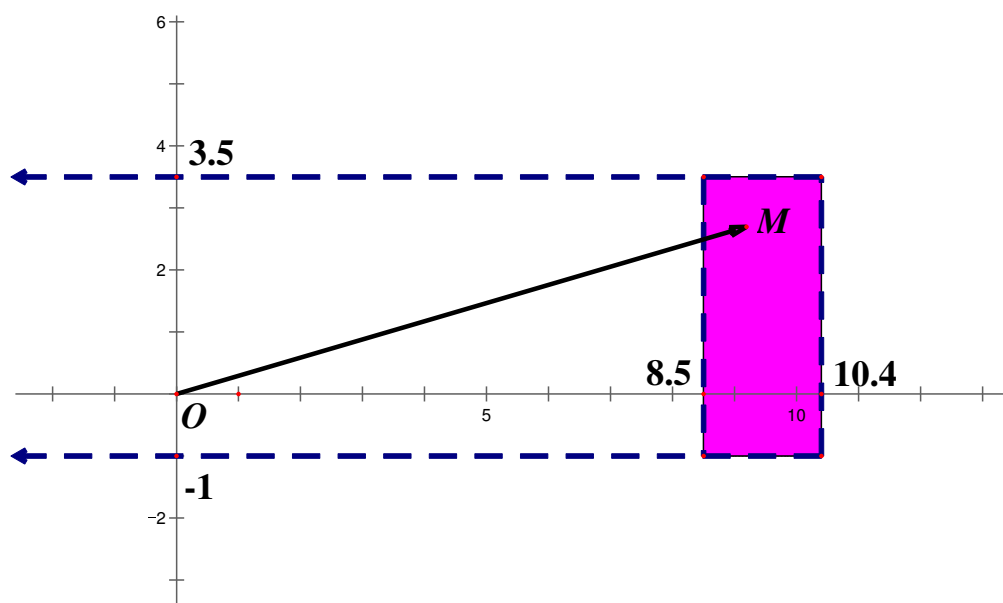


图 2.15: 雪球发射器的发射方向。O 为发射点，在如图所示矩形内随机取一点 M，则 OM 为发射方向。



### 2.3.6 烟花火箭

烟花神教主角。关于烟花火箭的信息请参考 wiki 和<https://www.bilibili.com/video/av5050255>。

### 2.3.7 传送机

一个传送机分为三个图格。传送机为前景物块，可以敲成半砖。传送机的工作机制中，三个图格分别有自己的传送区域（图 2.16）。



图 2.16: 三个图格的传送区域。如果图格被敲成下半砖，则传送区域下移半格，其他半砖形态传送区域不变。

当一根电线激活时，记录下该电线下第一个结算的传送机图格与最后一个结算的传送机图格<sup>5</sup>，然后将两个图格传送区域内的可传送目标互换，互换后它们的速度不变，位置相对于传送区域不变。当两个图格的传送区域有重合并且第一个图格不低于最后一个图格时无法传送<sup>6</sup>；当两个图格的传送区域有重合并且第一个图格低于最后一个图格时可以传送，此时两传送区域重叠部分属于第一个图格的传送区域（图 2.17）。

### 2.3.8 像素盒

像素盒可随意摆放，无需支撑块或背景墙。像素盒有十字状态的分线盒的分线效果。当一个电源激活时，该电源上的所有电线激活，此时如果像素盒上有横向电线激活且无纵向电线激活，那么像素盒熄灭；如果像素盒上既有横向电线激活又有纵向电线激活，那么像素盒点亮；如果无横向电线激活，那么像素盒不响应。

需要注意的是，像素盒的响应是对电源敏感的，即分别结算每个电源发出的信号，这与传送机对电线敏感不同。同时，与一般的光源在亮灭之间切换不同，像素盒响应总是调整到对应状态。

### 2.3.9 矿车轨道交叉点

交叉点上必须有两个方向的平滑轨道，那么这两个平滑轨道必定是一个覆盖另一个，此时激活交叉点会使两个平滑轨道的覆盖关系改变。需要注意的是激活交叉点可以产生的变化数量远远小于使用锤子可以产生的变化数量。

### 2.3.10 其他用电器

剩余的用电器在 wiki 之外的信息相当少，因此不专门介绍。它们是：雕像、烟花喷泉、烟花盒、泡泡机、呆萌气球机、派对中心、喷泉、八音盒、烟囱、天塔柱、广播盒、

<sup>5</sup>同一根电线上的结算顺序请参阅节 3.10

<sup>6</sup>这解释了为什么传送机不会自身传送。

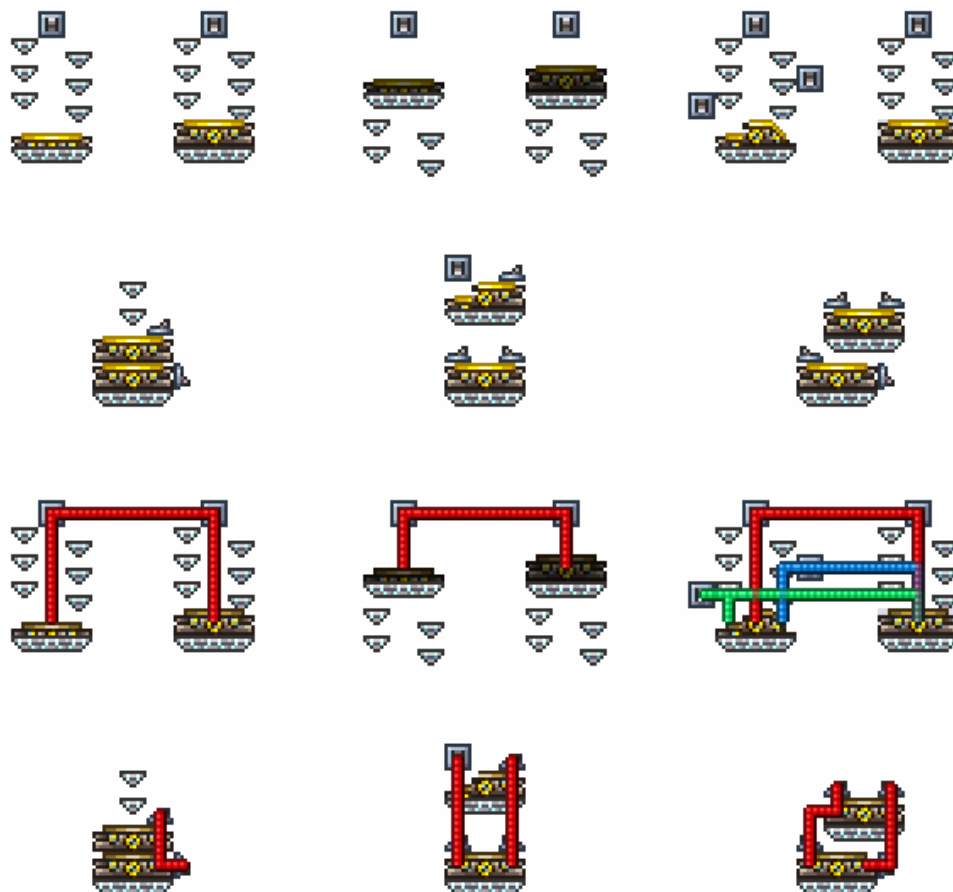


图 2.17: 左上装置：右边传送机比左边传送机传送区域高半格，因此玩家从左边传送到右边会高半格。中上装置：只要碰撞箱与传送区域有一点重叠，就可以传送。右上装置：传送机的三个图格各自有传送区域，从左边传送到右边，激活绿线会高半格，而激活蓝线红线不会。下方三个装置：激活上面开关不会传送，激活下面开关会传送；传送时只要玩家在下方传送机的传送区域内，就会被从下传到上；传送时只有玩家在上方传送机的传送区域内并且不在下方传送机的传送区域内，才会被从上传到下。

制动器、传送带、彩线灯泡、增速轨道。

## 2.4 逻辑门灯/逻辑门

逻辑门灯是用电器，简称为逻辑灯。逻辑灯还可以分为普通逻辑灯和故障逻辑灯。普通逻辑灯被激活时在开/关之间切换，故障逻辑灯被激活时状态被设定为“激活”（无图像效果）。逻辑灯必须堆叠在逻辑门上，我们说这些逻辑灯在该逻辑门上，该逻辑门在这些逻辑灯下。

逻辑门是电源，不同的逻辑门有不同的逻辑判定规则。如果一个逻辑门上没有故障逻辑灯，我们称其为普通逻辑门。在本节中讨论的逻辑门均为普通逻辑门。当一个逻辑门上的逻辑灯状态改变时，逻辑门会在下一个逻辑帧<sup>7</sup>进行逻辑判定并调整自己的亮灭状态。如果逻辑门的状态改变了，那么逻辑门会激活。

<sup>7</sup>逻辑帧的概念将在其有重要作用的时候详细解释，这里可以暂时认为是远小于 1 帧的时间。

### 2.4.1 二极管/换线器

在开始这一小节之前，先思考一个问题：如何用一个开关同时开启多于 4 个的 1 秒计时器并使它们独立工作？

要回答这个问题，首先需要明确的一点是，如果你不确定会发生什么，用一根电线连接两个计时器是非常不靠谱的。因为计时器既是电源又是用电器，连接在一起的两个计时器在下次激活的时候就会互相干扰，导致其中一个计时器将另一个计时器关闭。由于泰拉瑞亚中只有四种颜色的电线，如果要用一个开关同时开启多于 4 个计时器，那么必然有两个计时器要用同一种颜色的线激活。然而要让它们互不干扰，就意味着这两根线不能连接。同时又必须用大小只有 1 格的开关控制这两根线，那怎么办呢？

矛盾的根源在于，开关通过一根电线激活多个计时器后，经过一个计时器周期，计时器会通过开关上的电线互相干扰。要消除这种干扰，就要让信号既可以从开关传到计时器，又无法从计时器传到开关，即实现类似二极管的功能。

如图 2.18(a)所示，在与门上放一个逻辑灯就做成了一种二极管。用红线连接逻辑灯，蓝线连接与门（图 2.18(b)，图 2.18(c)），则激活红线会导致蓝线激活，而激活蓝线不会影响红线。根据与门的定义，当逻辑灯点亮时与门点亮，当逻辑灯熄灭时与门熄灭。红线激活时，逻辑灯在亮灭之间切换，从而逻辑门也在亮灭之间切换，激活一次蓝线。由于逻辑门不是用电器，所以激活蓝线不会影响红线。



图 2.18: ((b))((c)): 上面的开关能控制两个火把，而下面的开关只能控制一个火把

用这个原理可以轻松用一个开关开启很多 1 秒计时器并使它们互不干扰（图 2.19）。

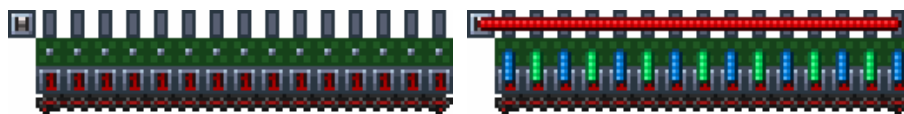


图 2.19

在另外一些时候，我们会用到换线器。例如我们已经预先做了一个计数器（计数器会在后面讲），它是使用蓝线输入的。同时我们做了一个驱动，该驱动使用红线输出。想用计数器来测量驱动的频率，就需要把计数器改成红线输入或者把驱动改成蓝线输出。当电路较复杂的时候，这个修改可能是非常困难的，这时就需要利用到换线器。由于这里驱动和计数器之间只是单向的信号传递，所以可以用图 2.18 中的二极管充当换线器，这样一来就解决了电线颜色不一致的问题。

### 2.4.2 1 秒延时器

打开 1 秒计时器后计时器每隔 1 秒激活一次电路。我们希望做一个 1 秒延时，打开 1 秒延时器后，经过 1 秒，1 秒延时器激活一次，然后关闭。

装置如图 2.20 所示。打开 1 秒计时器，经过 1 秒，计时器激活电路，然后逻辑灯被激活，进而逻辑门激活，把 1 秒计时器关闭。



图 2.20

### 2.4.3 改进自动门

上一节中我们使用逻辑感应器（玩家）实现了自动门的功能。这个自动门仍有缺点，那就是感应器的感应范围太大，导致玩家在很远的地方，甚至不想开门的地方（门的上方和下方）门也会打开。加重压力板的感应范围合适，但是当门两边同时站人时会导致门关闭。我们需要实现的功能是：当门的左边站人或右边站人时门开启。因此使用或门可以达到我们的要求。



图 2.21

在之前的电路分析中，我们都是使用的“激活”来分析。使用激活分析总是没错的，但是在有时不如状态分析来的简单。在图 2.21 中，所有的电源与用电器都是两状态的：加重压力板有踩下和弹起两个状态，逻辑灯、逻辑门有亮灭两个状态，制动器有实化和虚化两个状态。同时，所有电源会在状态改变时激活电路，所有用电器会在电路激活时状态改变。在这种情况下，电源和用电器的状态始终是同步的，例如逻辑门亮时制动器一定是虚化状态，逻辑门灭时制动器一定是实化状态；压力板踩下时对应的逻辑灯一定亮，压力板弹起时对应的逻辑灯一定灭。我们用这种思路来分析电路：门打开 = 逻辑门亮 = 两个逻辑灯至少有 1 个亮 = 两个压力板至少有一个被踩下 = 门附近有人。逻辑正确。

状态分析并非万能，它只适用于电源和用电器都是两状态切换的情况下。在多数情况下仍需要用激活分析。

### 2.4.4 绝对等价的逻辑门

泰拉瑞亚中共有六种普通逻辑门：包括与门、与非门、或门、或非门、异或门、异或非门。

这么多门的名字看起来十分头大，是不是必须熟悉每种门才能做电路呢？事实上，这六种门中有很多都是绝对等价的。两个装置绝对等价是指，如果把这两个装置都刷黑，那么没有任何方法可以测出它们的区别。如??，左边是与门，右边是与非门，分别放上数量和状态均相同的逻辑灯后，这两个门的表现完全一致。与门和与非门的唯一区别就是，对于相同输入，与门亮时与非门一定灭，与门灭时与非门一定亮。由于逻辑门在亮灭切换时会激活，与门和与非门总是同时激活。在泰拉瑞亚中，由亮到灭的激活和由灭到亮的激活是没有区别的，所以与门和与非门没有区别。同理，或门和或非门没有区别，异或门和异或非门没有区别。

现在我们已经把需要学习的范围缩小到了三个门：与门、或门、异或门。接下来我们看??，左边是与门，右边是或门，与门上全是亮灯，或门上全是灭灯。现在这两个门也是完全一致的。这是因为，它们上面的逻辑灯初始状态相反，对于相同输入，最终取值也相反。所以，与门亮  $\iff$  与门上的所有灯亮  $\iff$  或门上的所有灯灭  $\iff$  或门灭。这样一来与门和或门的亮灭状态永远是相反的，类似于前面的讨论，这里的两个门也没有区别。

现在只剩下两种门了：与门、异或门。这两个有没有可能绝对等价呢？不可能，因为与门满足结合律，而异或门不满足结合律。。读者在这里只需要知道，我们使用与门和异或门就足够了，其他普通逻辑门都可以用这两种门代替。

### 2.4.5 十进制数显

在上一节中，我们做了一个可以显示二进制数的数显。有了逻辑门后我们就可以做十进制数显。数显视输入不同有不同的做法，这里我们使用 4 位二进制数输入，要求二进制数在 0000~1001 时显示对应的十进制数 0~9，其他情况下显示屏熄灭。

我们选择使用七段线显示。首先用火把摆出七段线（图 2.22((a))），然后用七根电线分别控制每一段线（图 2.22((b))）。对于 0~9，a~g 分别有对应部分点亮表 2.3。

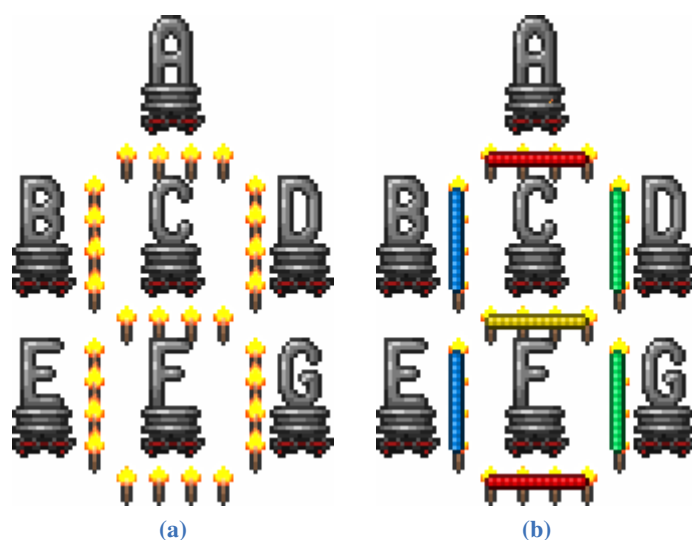


图 2.22

显示数值	0	1	2	3	4	5	6	7	8	9
点亮部分	abcefg	cf	acdeg	acdfg	bcd f	abdfg	abdefg	acf	abcdefg	abcdfg

表 2.3

然后我们来处理输入。我们需要把输入的四位二进制信息转换为 10 个数字中的一个，这个过程叫做解码。我们用图 2.23 中的电路进行解码。

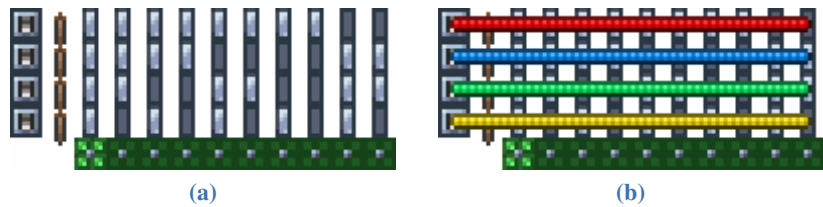


图 2.23: 十个与门从左到右依次代表 0~9，四个开关/火把从上到下依次代表 8,4,2,1。例如火把 4,2,1 点亮，则代表数字  $4+2+1=7$  的逻辑门点亮，其他逻辑门熄灭。

接下来就可以进行接线了。把十个逻辑门依次标号为 0~9。把七段线显示器置为 0 状态，然后依据表 2.3 接线：把 0 号逻辑门连到 abcefg，把 1 号逻辑门连到 cf，等等。遇到电线颜色冲突的时候要使用换线器，最终连好的电路如图 2.24。操作四个开关，显示屏即显示 0~9 的数字或者熄灭。之所以会这样，是因为每次数字改变的时候，原有数字的逻辑门熄灭，激活一次，使原数字对应的所有火把熄灭；新数字的逻辑门点亮，激活一次，使新数字对应的所有火把点亮。

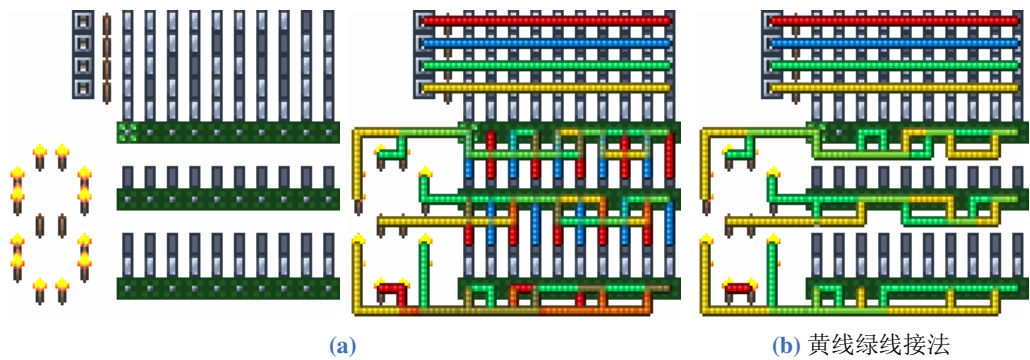


图 2.24: a 连接到 02356789，b 连接到 045689，c 连接到 01234789，d 连接到 2345689，e 连接到 0268，f 连接到 013456789，g 连接到 0235689。

## 2.5 故障逻辑门

如果任意的普通逻辑门上有至少一个故障逻辑灯，那么该逻辑门会变为蓝色，称为故障逻辑门。故障逻辑门是一个（不是一类）特殊的逻辑门，无论其下方的普通逻辑门是什么，故障逻辑门的性质是相同的。故障逻辑门与其上最下方的故障逻辑灯之间的普通逻辑灯为有效逻辑灯。当故障逻辑门上有一个故障逻辑灯被激活时，故障逻辑门会在下一个逻辑帧依一定概率激活并把其上所有故障逻辑灯状态设定为“未激活”。该概率等



于点亮的有效逻辑灯的数量除以所有有效逻辑灯的数量。请读者对照图 2.25 仔细揣摩这段话的描述。

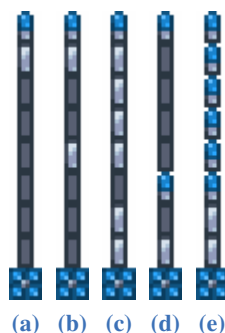


图 2.25: 当激活顶端的故障逻辑灯时, 故障逻辑门激活的概率分别是: ((a))1/7; ((b))2/7; ((c))6/7; ((d))1/2; ((e))1。

只有 1 个有效逻辑灯的故障逻辑门较特殊。因为该逻辑灯熄灭时对应的概率为 0, 故障逻辑门一定不激活; 当逻辑灯点亮时对应概率为 1, 故障逻辑门一定激活。使用只有 1 个有效逻辑灯的故障逻辑门可以进行电路控制。

### 2.5.1 换线器

在普通逻辑门那里我们已经讲了换线器的做法。简单来说, 换线器就是利用某些逻辑门遇输入必输出的特性。在实际应用中换线器的变种非常多。

单灯换线器只有一种, 就是任何一个普通逻辑门上加一个逻辑灯。双灯换线器因为有中间的逻辑门隔开, 输入和输出可以用同色的线。与门、异或门、故障门都可以做双灯换线器, 如??所示。

这三种换线器乍看起来又是等价的, 然而实际上不是。正因为它们不等价, 实际应用中可以利用它们的区别丰富我们的设计思路。

首先来看与门和故障门的区别。它们的最上面的逻辑灯激活时, 逻辑门都会激活。但是, 如果最上面的逻辑灯同时激活两次, 与门不会激活, 而故障门会激活一次, 对应的实验电路如??, 读者可以自己实验, 然后结合普通逻辑门和故障逻辑门的特性描述, 思考一下为什么会这样。异或门和故障门也有同样的区别。

再来看与门和异或门的区别。与门不抗干扰, 异或门抗干扰。什么意思呢? 看到??, 有一根与换线器无关的电线想要横穿换线器。如果是与门, 那么无论如何横穿, 这根线激活后换线器都会或多或少发生异常; 如果是异或门, 只要把这根线接到两个逻辑灯上, 那么这根线无论如何激活, 都不影响换线器的工作。读者可自行分析其中原因。可能你会问, 有谁会自找麻烦把一根线穿过去? 在实际应用中有时不得不这么做, 这时异或门就提供了便利。

### 2.5.2 置 1/置 0 电路

泰拉瑞亚中的逻辑电路有个非常大的缺点，就是赋值困难。在数电中，想要给电路赋值 0 或 1，只用连上对应电平电源即可。然而在泰拉瑞亚中，电线没有电平高低，只有激活，而激活只能改变逻辑灯状态。把一个逻辑灯赋值为 0，就需要在这个逻辑灯本来是 1 的情况下激活一次（或激活奇数次），在逻辑灯本来是 0 的情况下不激活（或激活偶数次）。如图 2.26 所示的电路可以做到这一点：当逻辑灯灭时，激活红线会将故障逻辑灯激活，因为逻辑灯灭，逻辑门不激活，逻辑灯保持灭。当逻辑灯亮时，激活红线会将故障逻辑灯激活，因为逻辑灯亮，逻辑门激活蓝线，逻辑灯熄灭。无论如何，激活红线都会使逻辑灯熄灭。



图 2.26

如果将逻辑灯与一个火把同步，就可以做到激活红线使火把熄灭。如果将逻辑灯与火把反向同步，就可以做到激活红线使火把点亮（图 2.27）。

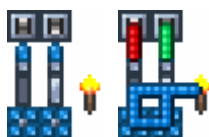


图 2.27: 左边开关使火把点亮（置 1），右边开关使火把关闭（置 0）。

### 2.5.3 递次电路

递次电路是使用频率非常高的电路。经典的递次电路如图 2.28 所示。当激活绿线时，一排故障逻辑灯被激活。但是由于只有第一个有效逻辑灯是亮的，只有第一个故障逻辑门激活，从而第一个有效逻辑灯熄灭，第二个有效逻辑灯被点亮。当再次激活绿线时，同理，第二个故障逻辑门激活，第二个有效逻辑灯熄灭，第三个有效逻辑灯点亮。依此进行，当反复激活绿线时，六个故障逻辑门依次激活并循环。将每个故障逻辑门接出一个电路，就可以实现六个电路依次运行并循环。



图 2.28

实际应用中的递次电路非常灵活。不仅故障逻辑门的数量可以随意变化，有效逻辑灯的亮灭、接线方式都可以视实际需求变化。递次电路的常见变种见小节 5.2.1。使用递

次电路时不要死板，要善于针对需求设计最合理的电路。

递次电路的使用非常广泛。它可以与十进制数显结合做成<https://www.bilibili.com/video/av22894193>中的十进制计数器，也可以做出<https://www.bilibili.com/video/av21009075>中的霓虹灯效果，还可以做出<https://www.bilibili.com/video/av6393957>中的回血回蓝大阵，等等等等等等。

递次电路的另一个典型用途就是降频。我们知道，传送带驱动的频率是 60Hz。现在我们需要 20Hz 的驱动，只需要利用循环长度为 3 的递次电路（图 2.29）。



图 2.29: 将黄线接出即得到 20Hz 的驱动，因为绿线每激活 3 次，黄线都激活 1 次。

在一些时候，比如上面提到的十进制计数器中，我们需要“清零”操作，即将递次电路复位到初始状态。使用置 0/置 1 电路很容易实现这一点（图 2.30）。但是这里想说明的是，对置 0/置 1 对象的理解需要非常灵活。

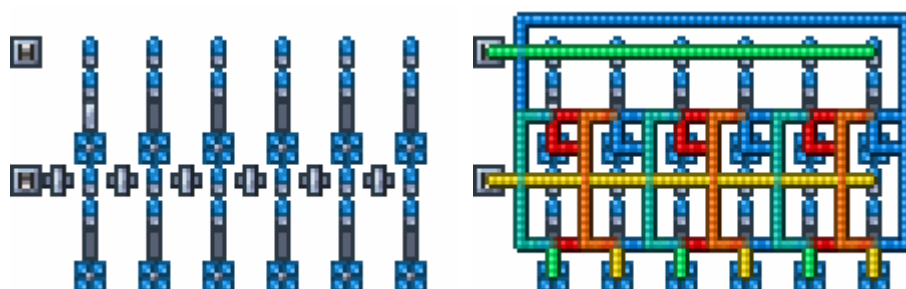


图 2.30: 上面的开关用来正常激活递次电路，下面的开关用来复位。除了横穿所有故障逻辑灯的绿线和黄线外，红线和蓝线用来进行递次电路中的循环并将上面有效逻辑灯的状态同步到下面，绿线和黄线用来置 0/1。多加一排故障逻辑灯是为了将颜色冲突的线分开。

是不是只有逻辑灯才有状态？在经典递次电路中，电线也可以有状态。如果我们把每根电线接到一个火把，就可以把火把的状态看作电线的状态。那么与其对逻辑灯置 0/置 1，不如直接对电线置 0/置 1，电路如图 2.31 所示。对电线进行赋值的前提是电线状态可以决定逻辑灯的状态。

将图 2.31 中下面一排复位电路穿插到上面递次电路的缝隙里，可以得到占用空间更小的版本（图 2.32）。

另外，偶尔我们可能需要用到双向递次电路，它的各种做法见小节 5.2.2。

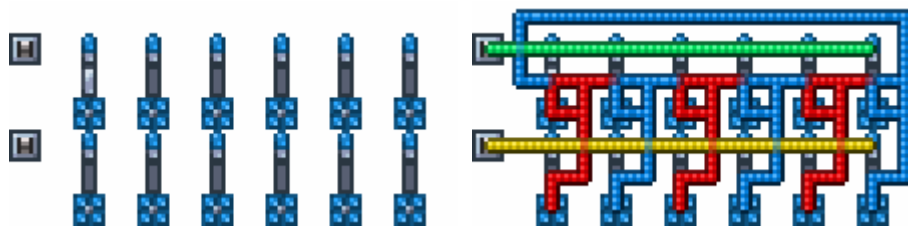


图 2.31: 上面的开关用来正常激活递次电路，下面的开关用来复位。



图 2.32: 上面的开关用来正常激活递次电路，下面的开关用来复位。

### 2.5.4 降频电路

我们已经讲过了如何利用递次电路降频。例如图 2.33((a))中，绿线激活奇数次时红线激活，偶数次时蓝线激活。事实上降频一半有更简单的方式。在图 2.33((b))中，激活绿线会激活故障逻辑灯并同时点亮有效逻辑灯，此时故障逻辑门会激活，红线激活；再次激活绿线会激活故障逻辑灯并同时熄灭有效逻辑灯，此时故障逻辑门不激活。图 2.33((b))也可以做到在绿线激活奇数次时红线激活，图 2.33((c))也可以做到在绿线激活偶数次时蓝线激活。一般说的降频电路就指这种使用一个故障逻辑门，把频率降低一半的电路。一般不要求精确频率时，使用降频电路比递次电路更节省空间。

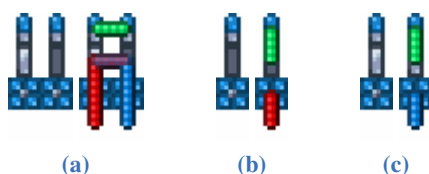


图 2.33

### 2.5.5 十进制计数器

在这一小节中我们将使用之前学过的模块来做一个四位带数显的十进制计数器。

计数器计数的对象是驱动或某个特定的信号源。信号源每激活一次，计数器的数字加 1。因为是十进制，所以满十进一，这提示我们使用循环为 10 的递次电路作为计数模块。因为计数器要有清零功能，所以递次电路要带上复位电路。根据数字的书写习惯，右边低位，左边高位，所以采用反向的递次电路并且将逻辑门稍微错位使接线更顺。将低位的递次电路最后一个逻辑门的输出接上高一位递次电路的输入即可完成进位功能（图 2.34），然后将最低位的递次电路输入接上驱动的输出即可以实现计数功能。

对递次电路熟悉的人通过递次电路上的逻辑灯已经可以读出数字：递次电路最右

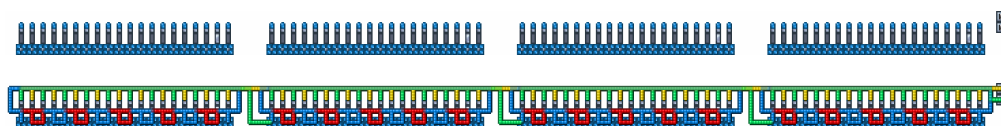


图 2.34: 绿线计数，黄线复位。最右边是个位，最左边是千位。

边的有效逻辑灯亮代表这一位是 0，最左边的有效逻辑灯亮代表这一位是 9。但是既然要面向不懂电路的用户，就需要把数字可视化，即加上数显。我们在之前已经做过一个十进制的数显，如果将那个数显的显示部分照搬过来，很容易发现显示的不对，这是因为两个数显的输入不同。之前做的数显，数字改变时显示器收到两个信号：第一个信号将之前的数字熄灭，第二个信号将新的数字点亮。而我们的递次电路的输出只是之前的数字。因为计数器中之前的数字可以唯一确定新的数字，所以我们可以直接让递次电路发出的信号激活从旧数字变成新数字需要激活的火把。另外，由于显示器是多个数字排列在一起，为了避免接线困难，应当先把数字上的线确定好，再往递次电路上接。如果选用图 2.35((a))所示的火把排布，由于数字间只空了一格，之前用过的七段线接法就不能用了。所以我们采用的另一种七段线接法图 2.35((b))，注意某些线有重叠。利用表 2.4 就可以完成接线（图 2.36）。注意到这里使用了故障逻辑门做换线器。

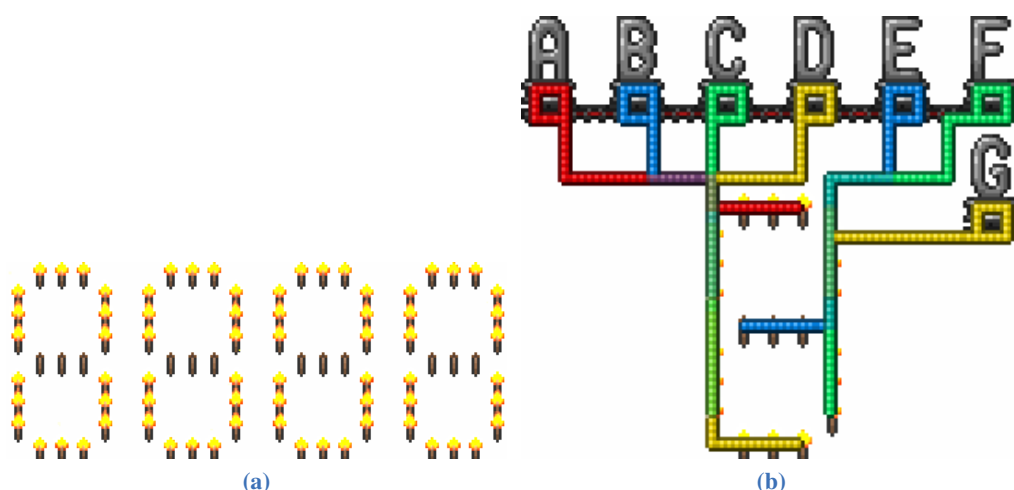


图 2.35

数字变化	0→1	1→2	2→3	3→4	4→5	5→6	6→7	7→8	8→9	9→0
激活部分	ad	abdeg	bcfg	abcd	acdf	bc	de	def	bc	bcef

表 2.4

## 2.5.6 骰子

在前面的例子中，我们仅使用了有一个有效逻辑灯的故障逻辑门的控制功能。在这个例子中我们来使用故障逻辑门的概率功能。我们的目标是做一个电路，该电路有一个开关和六个火把。当激活开关时，六个输出有且仅有一个点亮，且每个火把点亮的概率都为  $1/6$ 。

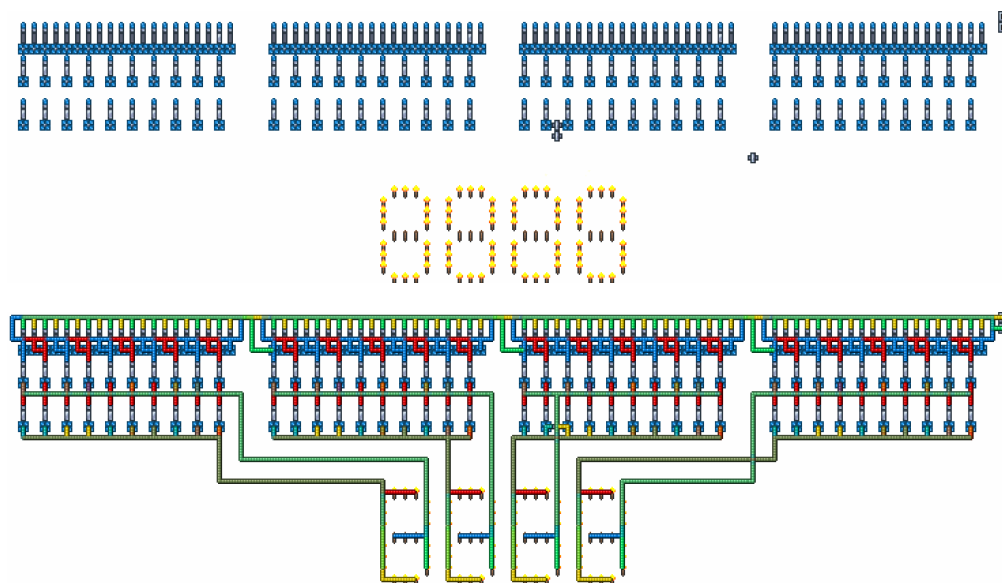


图 2.36

电路如图 2.37 所示。激活顶端黄线即可运行。下面很明显是递次电路，我们先分析上面部分，看看绿线会激活多少次。

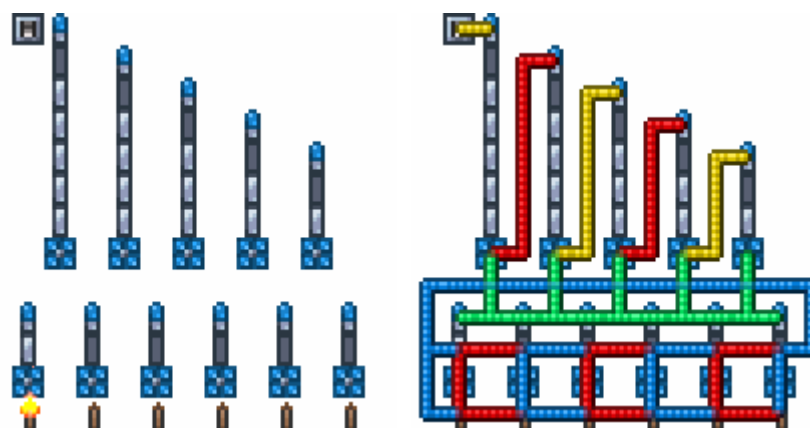


图 2.37

当顶端黄线激活时，第一个故障逻辑门有  $5/6$  的概率激活。只有第一个故障逻辑门激活的前提下，后面的故障逻辑门才有可能激活，也就是说，有  $1-5/6=1/6$  的概率一次也不激活。在第一个故障逻辑门激活的前提下，第二个故障逻辑门有  $4/5$  的概率激活。只有第二个故障逻辑门激活的前提下，后面的故障逻辑门才有可能激活，也就是说，有  $5/6*(1-4/5)=1/6$  的概率仅第一个故障逻辑门激活，此时绿线激活一次。依此类推，可以得到绿线有  $5/6*4/5*(1-3/4)=1/6$  的概率激活两次，有  $5/6*4/5*3/4*(1-2/3)=1/6$  的概率激活三次，有  $5/6*4/5*3/4*2/3*(1-1/2)=1/6$  的概率激活四次，有  $5/6*4/5*3/4*2/3*1/2=1/6$  的概率激活五次。也就是说，顶端黄线激活时，绿线等可能地激活  $0\sim 5$  次。

绿线等可能地激活  $0\sim 5$  次，经过下面的递次电路，就会导致点亮的火把等可能地循环右移  $0\sim 5$  个。无论激活顶端黄线之前哪个火把是亮的，这都意味着六个火把有等可能



的概率点亮。

## 第2章 思考题

1. 为什么只能由玩家激活的压力板会选用灰/棕/蓝/丛林蜥蜴这四种颜色？
2. 解释为什么如图 2.38 所示装置会在世界开启瞬间激活广播盒。该装置里的故障逻辑门有什么作用？



图 2.38

3. 制作一个装置，在每天入昼时显示讯息“日食正在发生...”，入夜时显示“血月正在升起...”。
4. 找出<https://www.bilibili.com/video/av5271577>中的延时装置。
5. 做一个显示当前月相的显示器，显示效果如图 2.39。

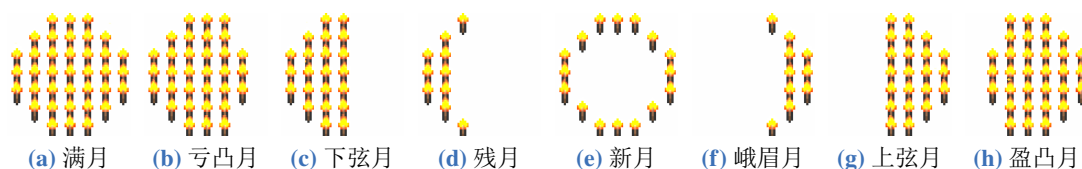


图 2.39

6. 实现<https://www.bilibili.com/video/av5271577> 2 分 57 秒处神庙尖刺的效果。
7. 设计如图 2.40 所示的二进制数显。

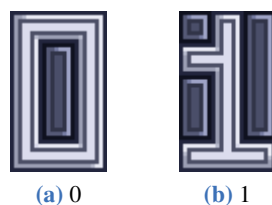


图 2.40

8. 补全十进制数显图 2.41 的接线。
9. 分析????的循环过程。



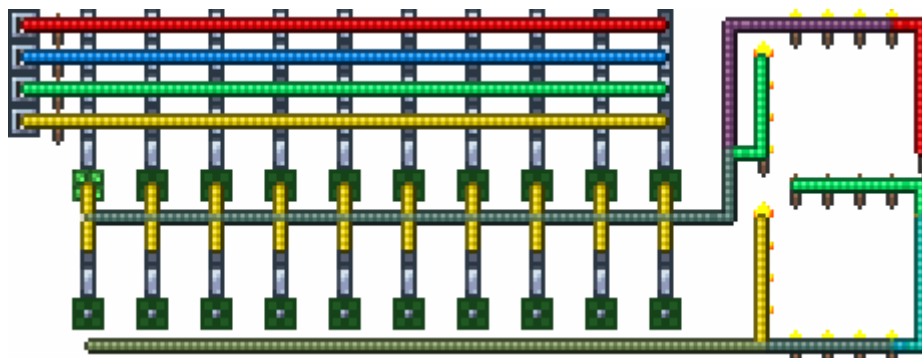


图 2.41

10. 为什么十进制带数显的计数器一般不会多于四位？
11. 利用降频电路和二进制数显制作一个多位的二进制计数器。
12. 做一个可以显示点数的骰子，六个点数效果如图 2.42。

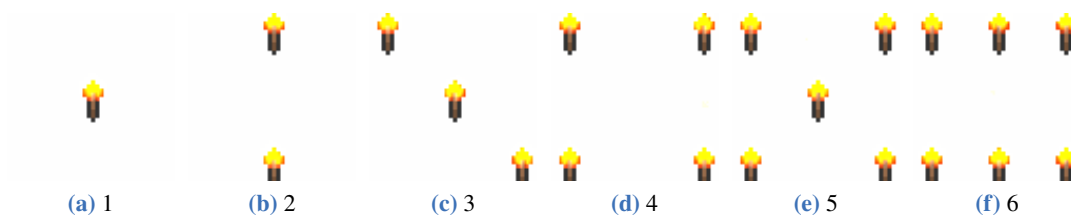


图 2.42

## 第 3 章 逻辑结算

在上一章中我们介绍了很多电路模块。这时候读者如果想当然地把这些电路模块组合起来，容易发生一些意想不到的问题，这也是上一章没有介绍过多复合电路的原因。在这一章中，我们将完善整个逻辑结算中的所有细节，确保可以预测任何复合逻辑电路的行为。

在一个逻辑电路中，每个电源和用电器可以接多根线，每根线可以接多个电源和用电器。当电路激活时，所有电源、电线、用电器都会按照一定顺序激活。

非常明显的结论是，电源激活后，电源上的电线激活，然后电线下的用电器激活。进一步的，如果电源  $A$  比电源  $B$  先激活，那么电源  $A$  上的电线也应当比  $B$  上的电线先激活；如果电线  $a$  比电线  $b$  先激活，那么  $a$  下的所有用电器也应当比  $b$  下的所有用电器先激活；如果逻辑门  $A$  上的逻辑灯比逻辑门  $B$  上的逻辑灯先激活，那么逻辑门  $A$  也应当比逻辑门  $B$  先激活。

以上的一些规则都是合理的，很容易理解。接下来的规则则是泰拉瑞亚特有的，需要进行记忆。

当一个电源激活后，其上的四色电线，按照红、蓝、绿、黄的先后顺序激活。

当一根电线激活后，其上各点按照距离电源由近到远的顺序激活。到电源距离相等的点，理解激活顺序需要理解图的广度优先搜索（BFS）算法，这里不详细解释。

对于涉及到逻辑门的电路，电路结算被分步进行。每一步中都进行逻辑门激活——电线激活——逻辑灯激活——逻辑门判断这四小步。逻辑灯被激活后，暂时不进行逻辑判断，等到所有电线结算完毕后，再统一进行逻辑判断。这每一步叫做**逻辑帧**。一个逻辑电路的结算过程中，反复运行每个逻辑帧中的四小步，直到没有逻辑门需要激活。

除了逻辑门以外的电源叫做**物理电源**。一个逻辑结算必须通过物理电源的激活启动。一个逻辑结算中一个逻辑门只能激活至多一次。如果它尝试激活第二次，那么激活失败并有白烟从逻辑门发出，称为**爆门**。逻辑门状态仍然会正常切换但是不会发出信号。爆门属于正常机制，是开发者有意为之，加以利用也可以起到作用。

### 3.1 用 MechScope 模组研究电路结算过程

MechScope 模组可以将电路结算过程可视化。其下载链接已经在节 1.1 中给出。安装该模组后，在游戏内可以看到电路的结算过程。

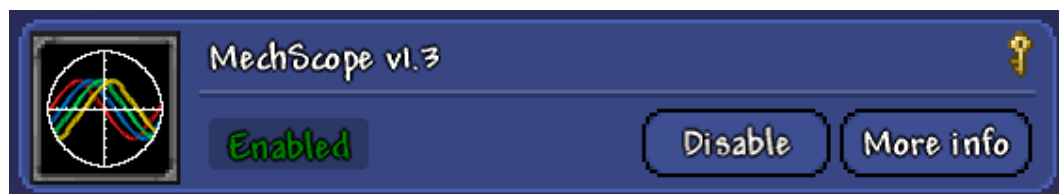


图 3.1: MechScope 模组

进入游戏后，打开设置——控制，可以看到 MechScope 的键位设置。默认键位为：

小键盘 1 Toggle（触发）

小键盘 2 Step（步进）

小键盘 3 Auto Step（自动步进）

小键盘 5 Settings（设置）

如图 3.2(a)所示，先做一个最简单的电路。在这个电路中，开关通过一根红线控制火把。



图 3.2

按下小键盘 1，可以看到光标右上角有一个黄点（图 3.3(a)），表示 MechScope 正在运行。

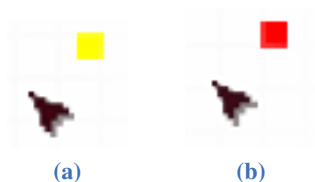


图 3.3

右键点击开关，此时电路会被暂停。开关上显示一个红框，电线所在格加了一层红色滤镜，分别表示激活的电源和电线（图 3.2(b)）。火把已经熄灭。光标右上角的黄点变成了红点（图 3.3(b)），表示有正在运行的电路。按一下小键盘 2 进行步进，电路结算完成，光标右上角恢复黄点。

了解了基本操作，我们来看设置菜单。按下小键盘 5，打开设置菜单（图 3.4）。前四项是单选，表示每次步进的幅度。Single 是逐格步进，Wire 是逐线步进，Source 是逐源步进，Stage 是逐逻辑帧步进。后六项是显示设置，仅影响可视化程度，这里不进行介绍，读者可自行尝试。最后一个数字是自动步进的周期/物理帧，这个值越小，自动步进越快。

我们的重心放在前四项设置，它们是有关于电路结算的，其中 Stage 是关于逻辑结算的。

### 3.1.1 逐格步进

设置菜单中选择逐格步进，然后做出图 3.5(a)所示的电路。打开 MechScope，右键点击开关，可以看到开关上出现了红框，表示开关作为电源，激活了它上面的红线（图 3.5(b)）。按下小键盘 2 步进，开关右边一格出现了红框，表示这一格激活（图 3.5(c)）。反复步进，红框一直向右移动直到火把上，把火把关闭。



图 3.4: MechScope 的设置菜单

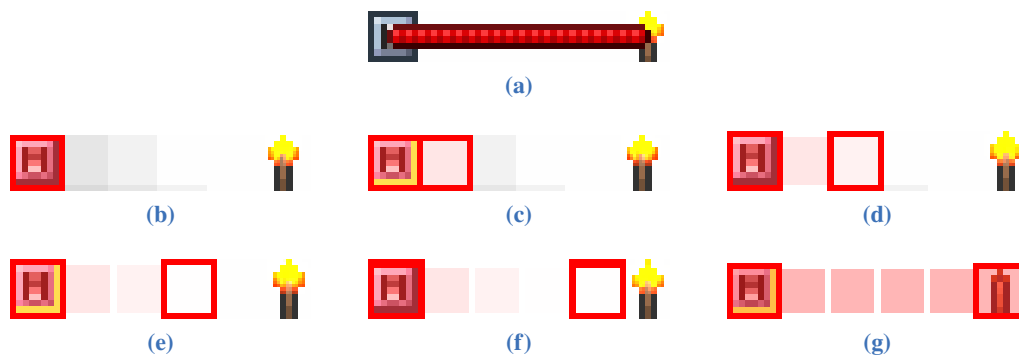


图 3.5

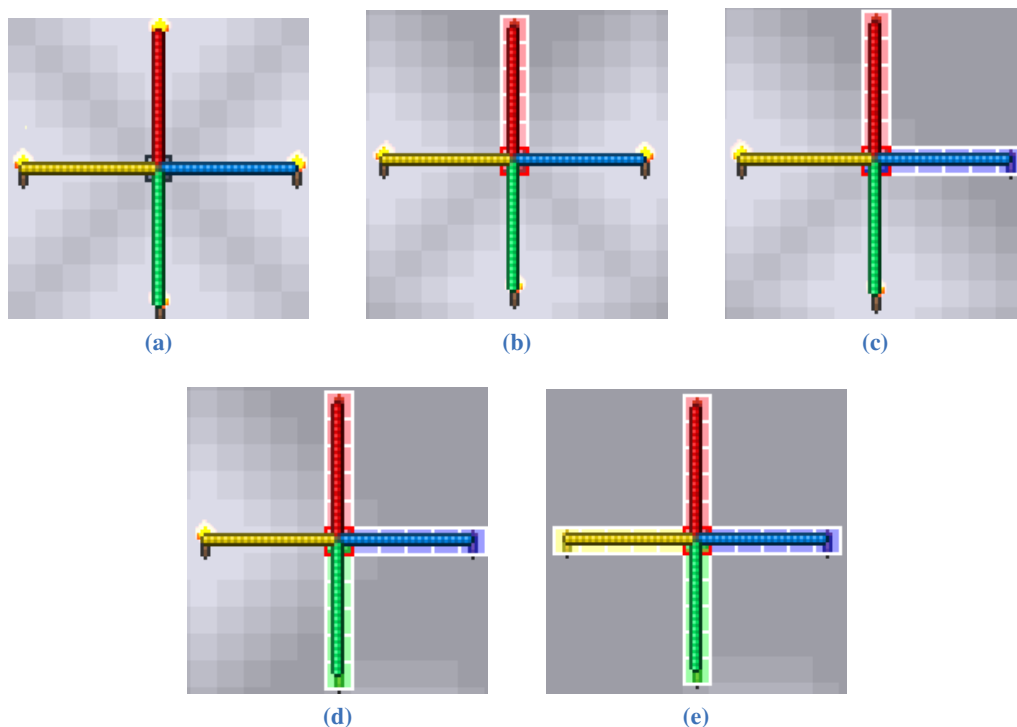


图 3.6

这个实验展示了单根电线上的激活顺序：电线由近到远依次激活。不过它只展示了单向传播的情况。对于复杂的情况，读者可以自己使用 MechScope 模组研究。一般情况下我们只需要知道距离电源近的先激活，距离电源远的后激活，这里的距离指沿电线到电源最短路径的长度。

### 3.1.2 逐线步进

设置菜单中选择逐线步进，然后做出图 3.6(a)所示的电路。在逐线步进模式下，每次步进会前进一整根电线。右键点击开关，可以看到红线激活（图 3.6(b)）；步进，蓝线激活（图 3.6(c)）；步进，绿线激活（图 3.6(d)）；步进，黄线激活（图 3.6(e)）。

这个实验展示了单个电源上不同电线的激活顺序：红线、蓝线、绿线、黄线依次激活。

### 3.1.3 逐逻辑帧步进

我们先跳过逐源步进，来看逐逻辑帧步进。电路如图 3.7(a)所示。在逐逻辑帧步进模式下，每次步进会前进一个逻辑帧。

我们一般把物理电源<sup>1</sup>激活时的逻辑帧记作第 0 个逻辑帧（图 3.7(b)），往后每个逻辑帧都进行逻辑门激活，电线激活，逻辑灯激活，逻辑门判断。在图 3.7中，从((b))到((g))分别是第 0 个逻辑帧到第 5 个逻辑帧，每个逻辑帧中都用加粗方框标出了激活的逻辑门，滤

<sup>1</sup>物理电源是除逻辑门以外的电源。

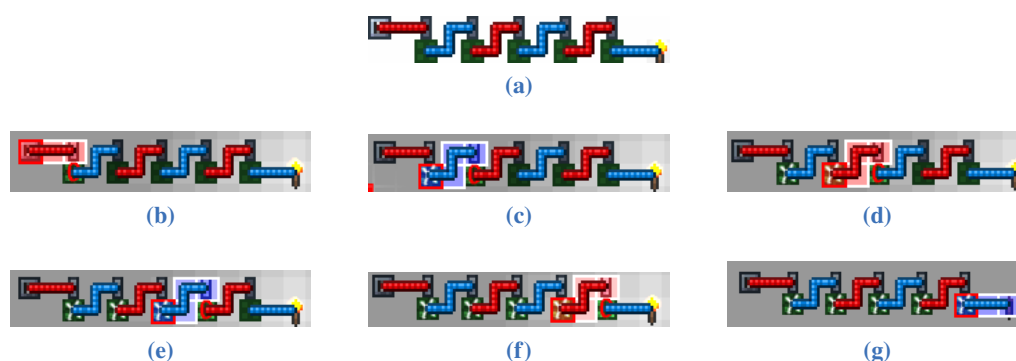


图 3.7

镜标出了激活的电线，“☒”标出了已经激活过的逻辑门，“”标出了在下一逻辑帧将要激活的逻辑门。

再来看一个稍复杂的电路（图 3.8(a)）。右键点击开关，进入第 0 个逻辑帧（图 3.8(b)）：开关作为电源激活，开关上的红线激活，两个逻辑灯激活，两个逻辑门进行判断，并计划激活（逻辑门上画了一个小的红圈）。

第 1 个逻辑帧（图 3.8(c)）。两个逻辑门作为电源激活，蓝线和绿线激活，三个逻辑灯及火把激活。其中火把改变状态，左右两个逻辑灯改变状态，中间的逻辑灯因为激活了两次，状态不变。三个逻辑灯下的逻辑门进行判断，其中左右两个逻辑门计划激活，中间的逻辑门不计划激活。

第 2 个逻辑帧（图 3.8(d)）。两个逻辑门作为电源激活，两根红线激活，四个逻辑灯及火把激活，全部改变状态。四个逻辑门计划激活。

第 4 个逻辑帧（图 3.8(e)）。四个逻辑门作为电源激活，蓝线激活四次，火把激活四次，状态不变。没有计划激活的逻辑门，逻辑结算结束。

### 3.1.4 爆门

来看图 3.9(a)，这是一个循环电路，看起来，红线在第 0 个逻辑帧激活，蓝线在第 1 个逻辑帧激活，红线又在第 2 个逻辑帧激活，蓝线又在第三个逻辑帧激活……如果真是这样，那么游戏会崩溃，因为泰拉瑞亚中，无论电路有多复杂，都必须要在一个物理帧内结算完毕。如果 1/60 秒没法结算完的话，物理帧会相应的加长，导致帧率降低。如果永远结算不完，那么游戏就会卡在一个物理帧中，无法进行任何操作。

泰拉瑞亚使用了一种傻瓜方法来规避这种死循环。一个逻辑门激活后，会有一个标记，在 MechScope 中会显示一个白色的“☒”。如果之后这个逻辑门还计划激活（MechScope 显示红色的“”），那么取消这个计划，并且播放爆门动画（冒烟）。在图 3.9中，爆门的行为发生在图 3.9(d)，即上面的逻辑门再次计划激活时。截图没能截下爆门的瞬间，读者可以自行尝试。

之所以称之为“傻瓜方法”，是因为它有副产物。换句话讲，对于不会死循环的电路也会发生爆门。在图 3.10中，下方的逻辑门在第 1 个逻辑帧激活，并被标记了“☒”。第 1 个逻辑帧中，蓝线激活，下方的逻辑灯状态改变，所以逻辑门又要计划激活，结果爆门。

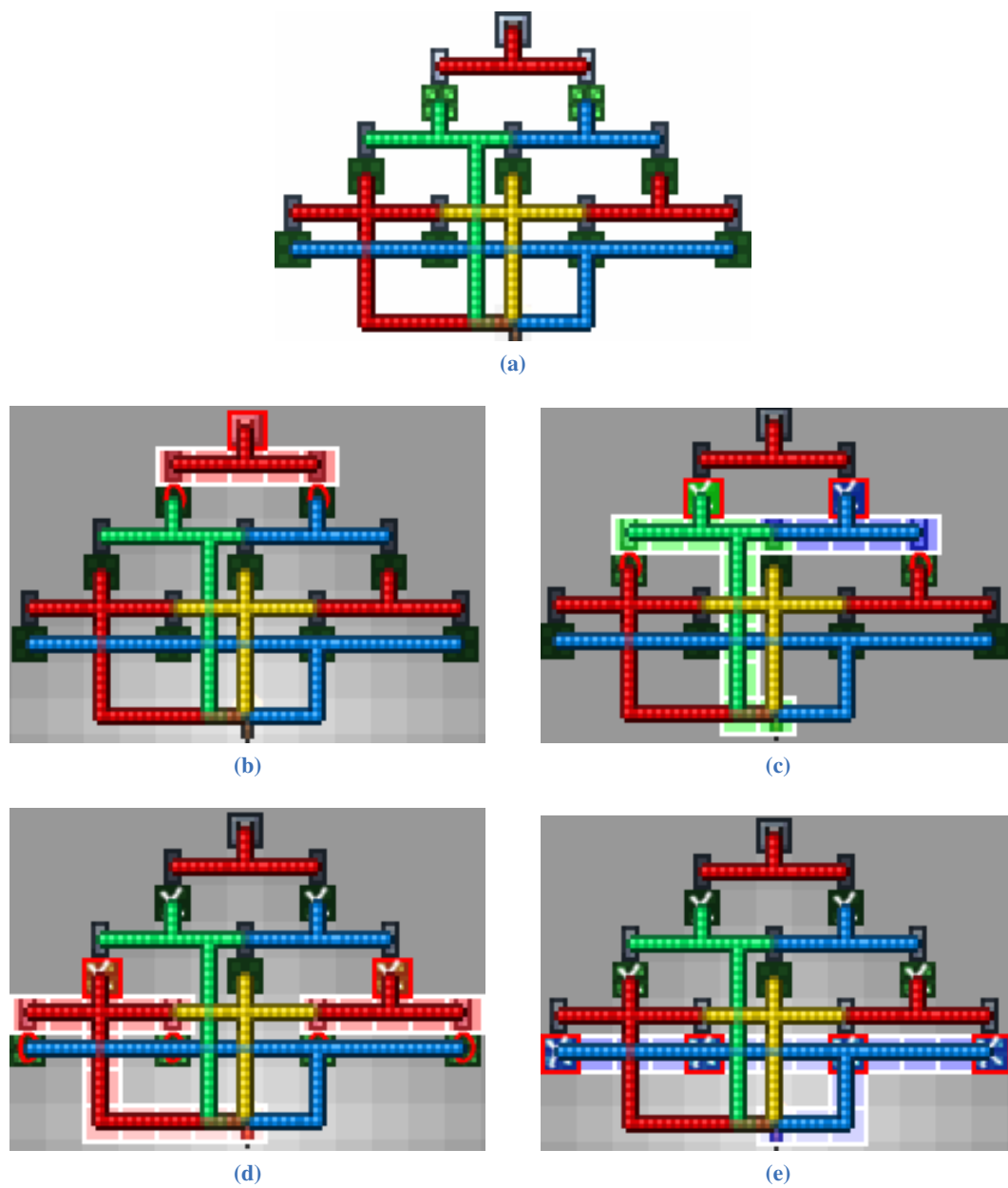


图 3.8

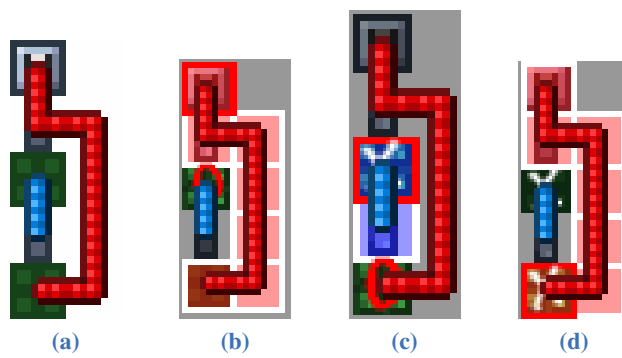


图 3.9



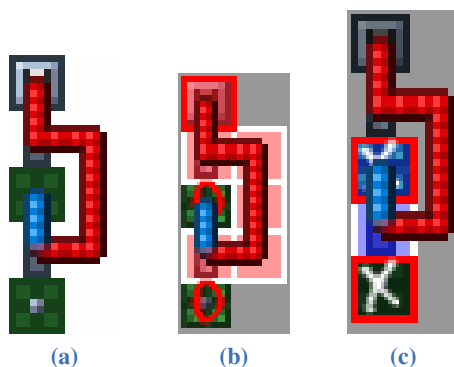


图 3.10

从上面两个例子中我们总结一下爆门的原因。一个逻辑门在一次电路结算中至多只会激活一次。如果尝试再次激活，那么激活失败并爆门。需要注意的是，逻辑门状态仍会照常改变。

在实际应用中，爆门经常导致电路 bug，这是因为我们往往希望逻辑门在多次状态切换时可以多次激活，但实际上不会。学会了预测爆门，就可以想办法避免爆门，甚至利用爆门。

## 3.2 逻辑延迟器

在连接多个逻辑电路模块时，要让它们正确地合作，就需要控制它们的运行顺序。不同电路模块从输入到输出经历的逻辑帧数量不同，所以简单的接线方式可能使得一些模块在不该工作时工作，打乱电路状态。使用逻辑延迟器可以推迟一个电路模块的运行逻辑帧，使得这个电路模块在需要运行的时候才运行。

上一章讲到的换线器就可以作为逻辑延迟器，因为输出激活比输入激活晚一个逻辑帧。如果将多个换线器首尾连接，就可以控制延迟的逻辑帧数量。

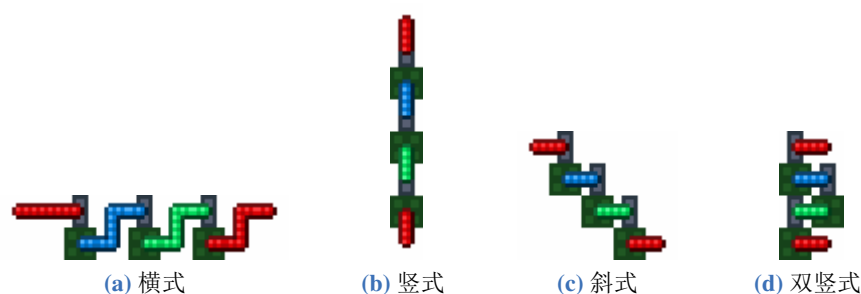


图 3.11: 逻辑延迟器的不同摆法，逻辑延迟均为 3 个逻辑帧。

### 3.3 普通逻辑门的逻辑同步

对于有数电基础的玩家，刚涉及泰拉瑞亚电路时会遇到各种各样的爆门。而通过各种教学视频入门的“外行”反而不容易遇到这样的问题。

爆门的原因是一个逻辑门在一次逻辑结算中激活多次。虽然故障逻辑门也可以发生爆门（下一节会讲），但是普通逻辑门才是爆门的重灾区。我们在使用普通逻辑门的时候，一般都是要进行简单的组合逻辑运算，比如几个与逻辑，几个异或逻辑的结合。这样的逻辑运算当然是可以进行状态判断的而不需要使用激活判断。可以进行状态判断的前提是，一根线连接的所有电源和用电器的状态要同步，换句话说就是，电源状态改变时，和电源连接的所有用电器状态也要改变。但是当爆门发生时，逻辑门状态改变却不发出信号，这样就会导致用电器状态不变，电路就无法进行状态判断了。不能进行状态判断的电路就没办法做组合逻辑。

举一个简单的例子。我们知道降频电路（[小节 2.5.4](#)）可以做二进制计数，而[小节 2.4.5](#)中的电路可以将二进制编码转成十进制显示。那么自然而然就会想到，把这两个电路连起来不就可以进行十进制计数了吗？

这里仅给出精简版的电路来说明这样做不对。如??，仅用一个降频电路，那么可以计四个数，用四个双灯与门进行解码，输出到四个火把。

开关第一次激活，仅红线激活，激活后各逻辑灯与逻辑门的状态如??，所以第一个门和第二个门激活，第一个火把熄灭，第二个火把点亮。开关第二次激活，红线与蓝线都激活，但是红线比蓝线早一个逻辑帧。红线激活后各逻辑灯与逻辑门状态如??，此时第一个门和第二个门激活，第一个火把点亮，第二个火把熄灭。蓝线激活后各逻辑灯与逻辑门状态如??，此时第一个门和第三个门激活，但是第一个门已经激活过了，所以爆门，只有第三个门激活，所以第三个火把点亮。到这里可以看出来，尽管逻辑门的状态是对的，即只有第三个逻辑门是亮的，但是火把状态不对，这是因为第一个门爆门后第一个火把少了一次激活。

导致这个爆门的原因是红线比蓝线早一个逻辑帧，解决起来也很简单，那就是在红线上做一个逻辑延迟，让红线和蓝线在同一个逻辑帧激活。

### 3.4 爆门

当一个逻辑门在逻辑结算中尝试激活多次时就会引起爆门。引起爆门的方式有多种（[图 3.12](#)）。

爆门只是一个机制，并非 bug，所以它也有可以应用的价值。例如，使用爆门可以将横式的逻辑延迟器体积大大缩小（[图 3.13](#)）。

### 3.5 状态表示与激活表示

泰拉瑞亚电路中的信号有两种表示方式：状态表示和激活表示。状态表示一般是显式的，即肉眼可以直接通过电路元件的状态读出其要表示的信息，例如亮表示 1，灭表示

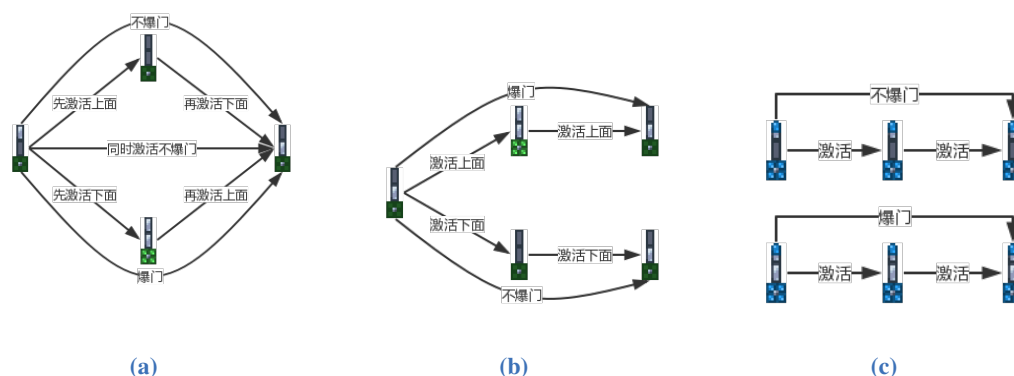


图 3.12: 可能引起爆门的三种原因。((a))一个普通逻辑门的不同逻辑灯在不同逻辑帧激活；((b))一个普通逻辑门的一个逻辑灯在多个不同逻辑帧激活；((c))一个故障逻辑灯在多个不同逻辑帧激活。



图 3.13: 改进的模式逻辑延迟器

0。激活表示是隐式的，激活表示 1，不激活表示 0。因为激活对于任何有显示效果的物品（像素盒除外）都只能改变其状态，因此想读出激活的信息，就需要在电路元件激活前和激活后的状态之间比较，状态变化的是 1，状态不变的是 0。这种读法显然是不方便的。

从电路运行角度，状态表示与激活表示经常需要搭配使用。普通逻辑门一般适用于状态表示，而故障逻辑门适用于激活表示。无论是为了产生可读性输出，还是为了在各电路模块之间传输信息，都会用到在状态表示与激活表示之间互换的方法。这一节中就将介绍将状态表示与激活表示互换的方法。

### 3.5.1 状态表示转为激活表示

一个故障逻辑门就可以做到将亮转变为激活，灭转变为不激活的功能。需要注意的是，仅用一个故障逻辑门的话，即使有效逻辑灯在一个逻辑结算中多次在亮灭之间切换，逻辑门也至多只能激活一次。所以更细分，有点亮立刻激活（图 3.14((b))）与事后激活（图 3.14((c))）两种造法。

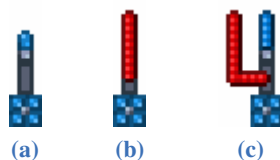


图 3.14: 红线控制有效逻辑灯状态。((b))有效逻辑灯点亮时立刻激活；((c))事后激活，在某个时刻激活蓝线可以将有效逻辑灯的亮/灭转换为故障逻辑门的激活/不激活。

点亮瞬间激活使用的就是降频电路，这里不作详细解释。事后激活即在某个时候（逻辑帧）中对于状态为 1 的激活，状态为 0 的不激活。

### 3.5.2 激活表示转为状态表示

激活转状态，需要将激活转变为点亮，不激活转变为熄灭。电路设计时需要特别注意“不激活”时的响应，因为按照电路原理，不激活时是不会有用电器响应的。如果要响应“不激活”，就必须在某个时间令电路执行不激活时的响应，即事前置 0（图 3.15）或事后置 0。

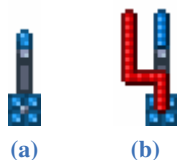


图 3.15: 无论红线是否激活，首先激活蓝线将有效逻辑灯置 0，随后红线激活则有效逻辑灯变为 1，否则有效逻辑灯为 0。

事前置 0 使用置 0 电路，在电路运行时先将火把置 0，然后若输入激活，则火把变为 1，否则火把保持为 0，就完成了激活转状态的功能。

事后置 0 既要使用置 0 电路也要使用置 1 电路。首先输入激活则激活置 1 电路，火把变为 1，然后当电路运行结束时激活置 0 电路将所有其他火把置为 0。要让置 0 电路不影响刚激活了的火把，需要使用额外的逻辑门记录本次逻辑结算中的激活状态，本质上还需要使用事前电路，因此这里不详细讨论。

### 3.5.3 事前电路与事后电路

事前电路，即在某个电路模块运行之前执行的电路；事后电路，即在某个电路模块运行之后执行的电路。把事前和事后的概念放在这一章，是因为控制它们执行的时机需要逻辑帧知识，并且有时需要在电路中添加逻辑延迟器来控制运行的时机。

从之前的讨论可以看出，状态转激活用事后电路较好，激活转状态用事前电路较好。

### 3.5.4 可以随机显示的十进制数显

上一章我们做的十进制数显输入还不够自由：要么需要输入二进制，要么需要连续数字。一个完美意义下的十进制数显应该接收 11 个输入，前十个输入激活则显示 0~9 的数字，最后一个输入激活则全部熄灭。

显示 0~9 的数字对应的是火把状态，而输入是激活。因此需要一个将激活转化为状态的电路。如图 3.16，在火把上加上事前置 0 电路。无论激活哪个开关，事前置 0 电路都会在第 0 个逻辑帧激活，在第 1 个逻辑帧生效。显示电路在第 0 个逻辑帧激活，在第 1 至 2 个逻辑帧生效。

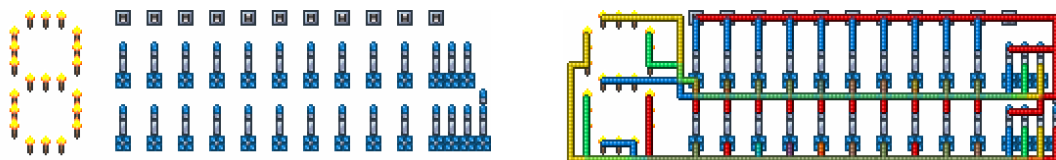


图 3.16

## 3.6 密码门

## 3.7 随机分两组

上一章我们学习了骰子的电路，即从  $n$  个元素里随机选取出 1 个。这一节中将介绍从  $n$  个元素里随机选取  $m$  个（或将  $n$  个元素随机分成两组，每组的元素数量分别为  $m$  和  $n-m$ ）的电路。要理解本节内容需要有高中的概率知识。我们以随机 8 选 4 为例。

首先考虑需要用到哪些概率的故障逻辑门。

- 第一个元素有  $4/8$  的概率分到第 1 组。
- 第一个元素的分组确定后就可以来给第二个元素分组，这时有两种情况。如果第一个元素分到了第 1 组，那么第二个元素有  $3/7$  的概率分到第 1 组，否则该概率为  $4/7$ 。
- 同理，如果前两个元素全部分到了第 1 组，那么第三个元素有  $2/6$  的概率分到第 1 组；如果前两个元素中只有 1 个分到了第 1 组，那么第三个元素有  $3/6$  的概率分到第 1 组；如果前两个元素全部分到了第 2 组，那么第三个元素有  $4/6$  的概率分到第 1 组。
- 如果前三个元素有  $3/2/1/0$  个分到了第 1 组，那么第四个元素有  $\frac{1}{5}/\frac{2}{5}/\frac{3}{5}/\frac{4}{5}$  的概率分到第 1 组。
- 如果前四个元素有  $4/3/2/1/0$  个分到了第 1 组，那么第五个元素有  $\frac{0}{4}/\frac{1}{4}/\frac{2}{4}/\frac{3}{4}/\frac{4}{4}$  的概率分到第 1 组。
- 如果前五个元素有  $4/3/2/1$  个分到了第 1 组，那么第六个元素有  $\frac{0}{3}/\frac{1}{3}/\frac{2}{3}/\frac{3}{3}$  的概率分到第 1 组。
- 如果前六个元素有  $4/3/2$  个分到了第 1 组，那么第七个元素有  $\frac{0}{2}/\frac{1}{2}/\frac{2}{2}$  的概率分到第 1 组。
- 如果前七个元素有  $4/3$  个分到了第 1 组，那么第八个元素有  $\frac{0}{1}/\frac{1}{1}$  的概率分到第 1 组。

乍一看这就需要 24 个故障逻辑门，还没有算上控制电路。如果  $m$  和  $n$  再稍微大点，就是天文数字了。仔细观察可以发现，每个元素的分组概率虽然有很多个可能值，但是这些概率都有两个特点：分母相同；分子与之前元素的分组情况有非常简单的关系。这就提醒我们使用之前元素的分组状态直接修改后续故障逻辑门的有效逻辑灯的亮灭，从而修改后续元素的分组概率。

用于概率的故障逻辑门组如图 3.17 所示。

激活第一个故障逻辑灯，第一个故障逻辑门有  $4/8$  概率激活。我们假设逻辑门激活

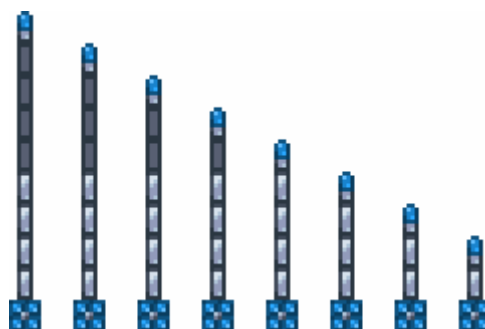


图 3.17

代表对应元素被分到第 1 组。当第一个故障逻辑门未激活时，我们希望第二个故障逻辑门保持当前的  $4/7$  的概率；当第一个故障逻辑门激活时，我们希望第二个故障逻辑门的概率被修改为  $3/7$ ，即有一个点亮的有效逻辑灯被熄灭。总而言之，只要第 1 组和第 2 组都没有排满，之前的逻辑门每激活一次，都会使后面逻辑门的概率降低。当第 1 组或第 2 组已经分到 4 个元素时，后面逻辑门的概率都保持在 0 或 1 不变了。

由于后面逻辑门的概率降低是根据之前逻辑门激活的次数确定的，所以使用递次电路完成这个计数功能。递次电路以所有用于概率的故障逻辑门作为输入。递次电路激活 1 次，表示有一个元素分到了第 1 组，将从下到上第 4 排有效逻辑灯熄灭，使后面分组的分子至多为 3；递次电路激活 2 次，表示有两个元素分到了第 1 组，再将从下到上第 3 排有效逻辑灯熄灭，使后面分组的分子至多为 2；...；递次电路激活 4 次，表示已经有 4 个元素分到了第 1 组，将最下面一排有效逻辑灯熄灭，此时所有有效逻辑灯熄灭，代表后面的概率全部为 0，即剩余所有元素都被分到第 2 组。为接线方便，采用斜向的递次电路（图 3.18）。

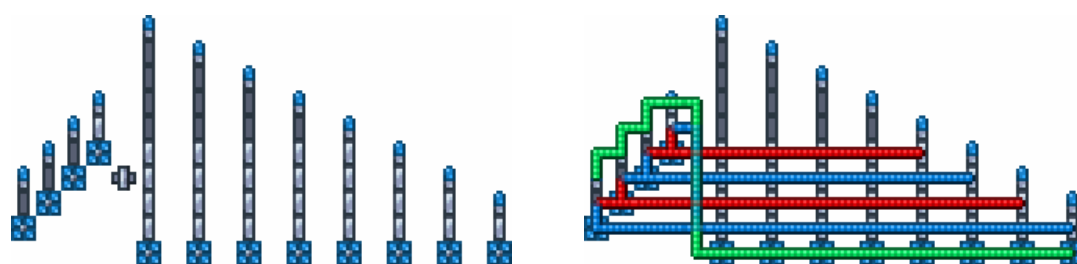


图 3.18

为重复使用，复位电路也是必须的，即将八个用于概率的故障逻辑灯全部激活后将电路恢复到初始状态。全部激活后，递次电路一定被激活了 4 次，所以递次电路已经自动复位；有效逻辑灯一定全灭，复位电路只需要将需要的有效逻辑灯全部点亮即可（图 3.19）。

在上面的电路中，我们需要依次从左到右点击八个开关，对应的故障逻辑门激活代表对应的元素被分到第 1 组<sup>2</sup>。分完以后需要点击第九个开关复位。理想的电路当然是希望只需要点一次开关就自动完成所有功能。整个电路的运行顺序应该是：第一个故障逻辑灯激活，第一个故障逻辑门（可能）激活后，递次电路激活，修改有效逻辑灯，然后第

<sup>2</sup>注意这里用激活表示分组，因此如果使用火把显示，需要额外的置 0 电路。



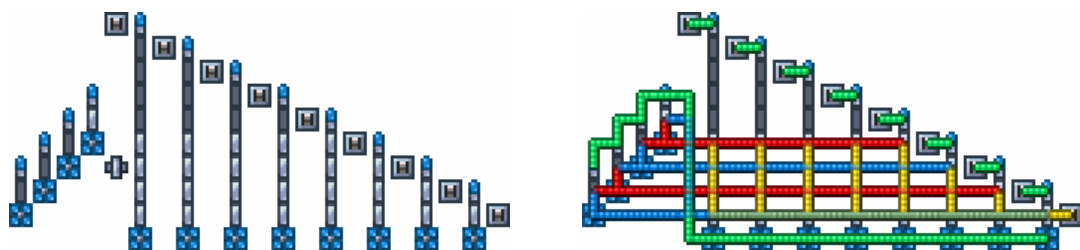


图 3.19

二个故障逻辑灯激活，……八个故障逻辑灯激活后，复位电路激活。因此每两个相邻的故障逻辑灯之间需要有逻辑延迟，最后一个故障逻辑灯和复位电路之间有逻辑延迟。接下来需要计算至少需要延迟几个逻辑帧。

当一个用于概率的故障逻辑灯在第  $k$  个逻辑帧激活后，该故障逻辑门在第  $k+1$  个逻辑帧激活，并激活递次电路；递次电路在第  $k+2$  个逻辑帧激活，并将对应的有效逻辑灯关闭。因此下一个故障逻辑灯激活时机不能早于第  $k+2$  个逻辑帧，否则概率还没有修改。也就是说，相邻两个故障逻辑灯之间的逻辑延迟至少为 2 个逻辑帧。

当最后一个用于概率的故障逻辑灯在第  $k$  个逻辑帧激活后，该故障逻辑门在第  $k+1$  个逻辑帧激活，并激活递次电路；递次电路在第  $k+2$  个逻辑帧激活并关闭对应的有效逻辑灯。由于递次电路和复位电路都只是对有效逻辑灯做操作，因此谁先谁后对电路运行结果没有影响，只需要复位电路运行时不干扰到故障逻辑门的概率判断即可。因此复位电路激活时机不早于第  $k+1$  个逻辑帧，即复位电路至少比最后一个故障逻辑灯晚一个逻辑帧激活。

捋好这些逻辑后就可以接线了。如图 3.20，在每两个相邻的故障逻辑灯之间加两个换线器用来构造 2 个逻辑帧的延迟；在最后一个故障逻辑灯和复位电路之间加一个换线器用来构造 1 个逻辑帧的延迟。火把上加了事前置 0 电路用来将激活转换为点亮。

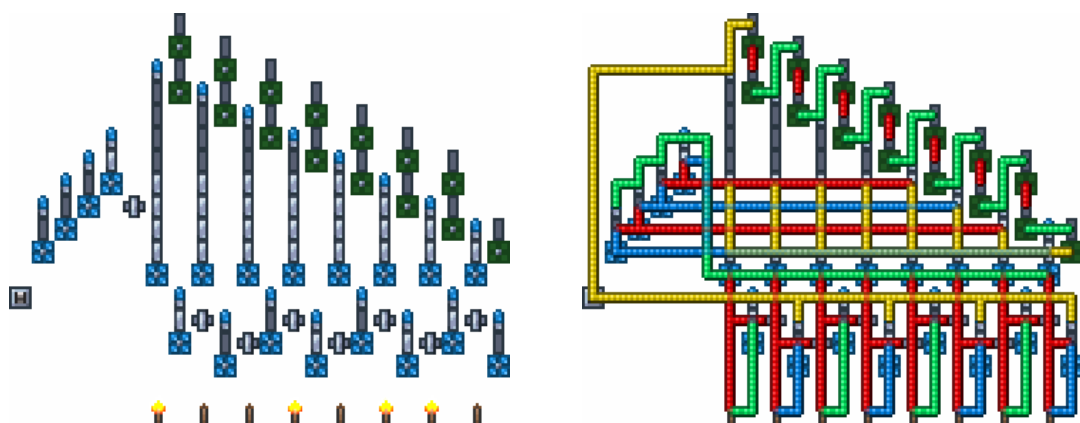


图 3.20

这里只是举了随机 8 选 4 的例子。随机  $m$  选  $n$  电路除了逻辑门和逻辑灯数量以外，原理和构造完全一样。



## 3.8 随机分三组

上一节中我们介绍了随机分两组的电路，这一节介绍随机分三组，其原理可应用于随机分多组。这节中的电路需要把 9 个元素分成 3+3+3 三组。

### 3.8.1 核心模块

要把 9 个元素均分为三组，只需要先分为 6+3 的两组，再把有 6 个元素的组分成 3+3 两组。电路如图 3.21 所示。上面的分组电路，将被分到 6 的一组激活。每有一个元素被分到 6 的一组，递次电路就将下面的 6 选 3 的电路的对应故障逻辑灯激活。如果这个逻辑门又被激活了，那么这个元素被分到第 1 组，否则分到第 2 组。如果一开始 9 选 6 都没有激活，那么分到第 3 组。由于 9 选 6 中相邻两个故障逻辑门已经设置了 2 逻辑帧的延迟，因此下面的 6 选 3 相邻的故障逻辑门间延迟已经至少是 2 个逻辑帧，因此不再需要调整。受电线颜色限制，6 选 3 的复位电路与 9 选 6 略有不同。

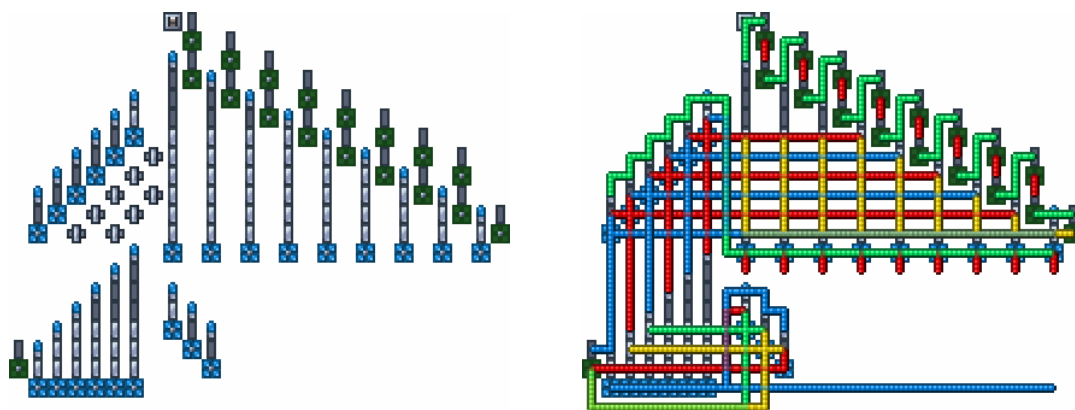


图 3.21

通过识别该核心模块发出的激活信号，已经可以判断出分组情况：如果 9 选 6 的某个故障逻辑灯未激活，那么该元素被分到第 3 组；如果这个故障逻辑灯激活了，但是两个逻辑帧后，6 选 3 下方的蓝线未激活，说明该元素被分到第 2 组；如果这个故障逻辑灯激活后两个逻辑帧，6 选 3 下方的蓝线恰好激活，那么该元素被分到第 1 组。

如果要将分组可视化，就需要做一个判断：6 选 3 下方的蓝线是否在 9 选 6 的某个故障逻辑灯后 2 逻辑帧激活？这种精确到逻辑帧的判断需要通过下面的激活与门来实现。

### 3.8.2 激活与门

激活与门是电路中用于信号处理的模块。一般的与门的判定对象是逻辑灯的状态，而激活与门的判定对象是输入的激活。即：当且仅当两个输入同时激活时，输出才激活。这个功能看起来简单，但是能完全实现描述的功能的模块目前还没有做出来。目前能做出来一些具有类似功能的模块，这里仅提供两个最简单的：两个输入在一个逻辑结算中都只输入一次恰好在同一个逻辑帧输入，那么输出在下一个逻辑帧激活；一个输入在一个

逻辑结算中只激活一次，并且在激活的同一个逻辑帧中，另一个输入恰好激活。这两个模块的电路图如图 3.22 所示。



图 3.22: 只有激活蓝线和红线重合处的开关，火把才会响应。

先看图 3.22(a)。当红线在第  $k$  个逻辑帧第一次激活时，左上与门上的逻辑灯激活，右与门上的第一个逻辑灯点亮；在第  $k+1$  个逻辑帧，左上与门激活，使得左上与门上的逻辑灯再次激活，右与门上的第一个逻辑灯熄灭；在第  $k+2$  个逻辑帧，与门尝试激活，但是因为已经在第  $k+1$  个逻辑帧激活过，所以与门爆门，不再激活。也就是说，右与门上的第一个逻辑灯在第  $k$  个逻辑帧点亮，并在第  $k+1$  个逻辑帧熄灭。因此，仅当两个输入在同一个逻辑帧激活时，右与门才可能点亮并激活，随后右与门的第一个和第三个逻辑灯在下一个逻辑帧就熄灭，右与门熄灭，尝试激活并爆门。

再看图 3.22(b)。对于红线的分析同理。蓝线每次激活时，如果在同一个逻辑帧中有效逻辑灯是亮的，那么故障逻辑门激活，否则故障逻辑门不激活。在前一个模块中蓝线只能激活一次，是因为左下与门已经爆门。在后面这个模块中，因为故障逻辑灯的激活次数没有限制，所以蓝线可以任意激活。

在这两个装置中有一个看起来很奇怪的接线方式：将换线器中的与门和逻辑灯直接连接。这个连法使得逻辑灯在点亮后一个逻辑帧立即熄灭，从而可以实现精确到逻辑帧的测量。利用这个原理可以实现很多类似装置，具体放在思考题中。

### 3.8.3 信号处理

现在我们回到随机分组的电路。我们已经知道，当 9 选 6 的某个故障逻辑门激活后，如果在两个逻辑帧后，6 选 3 的输出恰好激活，那么对应元素被分到第 1 组。我们可以通过激活与门来实现这个功能。

选好了信号处理方式，再选择可视化方式。火把只有两个状态，没法表示分成三组的情况，因此只能使用多个火把或者使用彩线灯泡。彩线灯泡显示更直观，所以这里选用彩线灯泡：使用 9 个彩线灯泡，灯泡显示红色、蓝色、绿色分别代表分到 3 组。

由于 9 个灯泡的信号处理电路完全相同，这里只讲一个的原理。我们做的这个信号处理电路有两个输入（红线和蓝线）。如果红线没有输入，那么灯泡为红；如果红线输入且蓝线在之后的第二个逻辑帧没有输入，那么灯泡为蓝；如果红线输入且蓝线在之后的第二个逻辑帧输入，那么灯泡为绿。另外，红线在逻辑结算中至多输入一次，蓝线一定会输入三次。由于没有输出时灯泡为红，所以可以使用置 0/置 1 电路将灯泡置为红色。当红线激活时，将灯泡的红色熄灭，蓝线点亮。当蓝线在两个逻辑帧后激活时，将蓝色熄灭，绿色点亮。电路如图 3.23 所示。在红线上接两个换线器，这样判定就成了当红线和蓝线在同一个逻辑帧激活时蓝色熄灭，绿色点亮，从而应用激活与门。左右两个故障逻

辑门用于事前将彩线灯泡置为红色<sup>3</sup>。

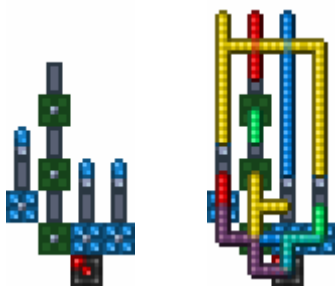


图 3.23: 上面的红线、蓝线、黄线分别为红线输入、蓝线输入、复位。

### 3.8.4 电路优化

把信号处理装置接到灯泡上和核心模块上时有个问题：核心模块输出的空间间距比较窄（每个输出的宽度为 2）；为美观起见彩线灯泡之间的距离也要尽量窄。然而上面的信号处理装置宽度为 4，所以要么接线会比较丑（图 3.24），要么就只能将核心模块和彩线灯泡之间的距离强行拉宽。所以对于信号处理模块的空间压缩是有必要的。

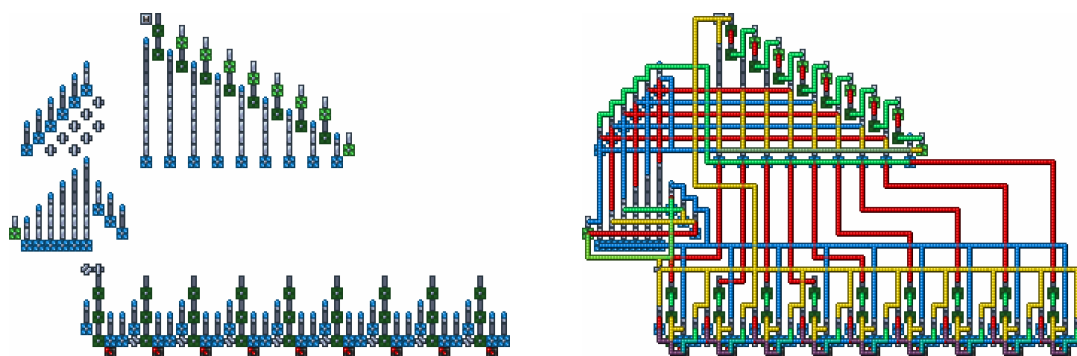


图 3.24

图 3.25和图 3.26分别是占用宽度为 2 和 3 的解，还不确定是否是对应宽度的最优解。占用宽度越小，占用高度越大<sup>4</sup>。电路优化没有固定套路，需要通过大量的实践与尝试总结经验，因此这里省略优化过程 1 万字，仅放出电路图供参考。

## 3.9 二进制加减法计算器

根据计算机习惯，我们在这一节中做一个八位二进制加减法计算器。在上一章中我们学习了全加器的做法。有了全加器以后就可以做加法器，利用补码可以做减法器。

<sup>3</sup>注意，尽管故障逻辑门上接了两种颜色的电线，只有一种颜色的电线接到了有效逻辑灯。

<sup>4</sup>有得选的话没人会用占用宽度和高度都大的电路。

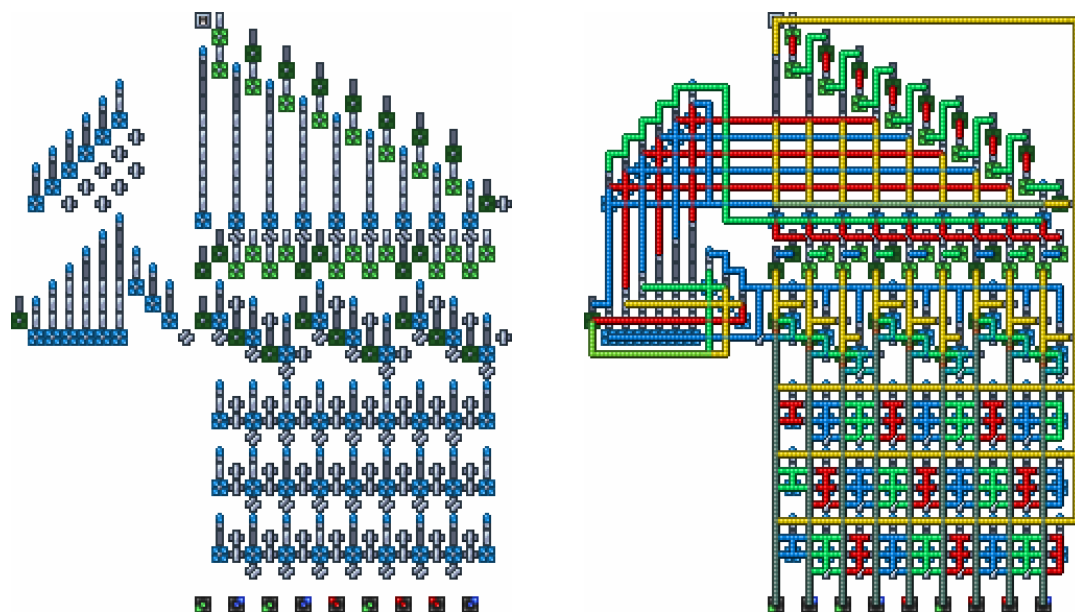


图 3.25

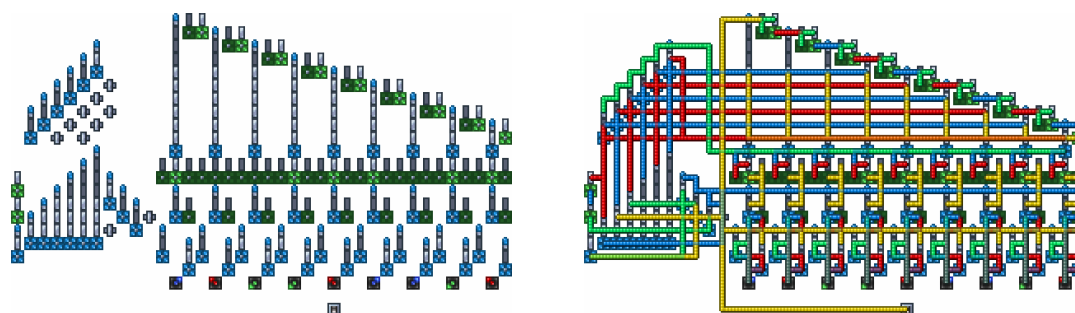


图 3.26

### 3.9.1 加法器

原理上我们选用最简单的逐位进位加法器，即计算和每一位时，接收三个输入：两个加数同一位的值，低位进位值；产生两个输出：和的这一位的值，向高位的进位。全加器就是实现这一功能的模块。

把 8 个全加器连接起来就可以做成一个 8 位的加法器（图 3.27）。每个全加器的三个输入分别接两个加数和低位全加器的进位输出，和数输出接到表示和数的火把，进位输出接到高位全加器的输入。

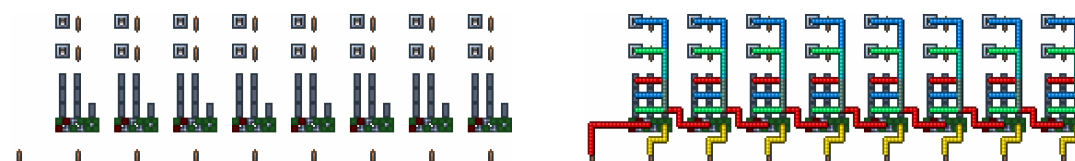


图 3.27: 第一排火把表示第一个加数，第二排火把表示第二个加数，最下面一排火把表示和。黄线输出使用了换线器。

需要注意的是，这个加法器作为一个逻辑模块，使用时要注意其运行顺序。当最低位有输入时，最低位的全加器在第 1 个逻辑帧结算，然后剩余全加器从右到左依次在第 2~8 个逻辑帧结算。所以作为逻辑模块时，输入要保证最低位先输入，然后每位依次延迟 1 个逻辑帧输入，否则可能引发爆门。

另外，这个加法器的最低位全加器没有进位输入，因此如果只做加法的话可以去掉一排逻辑灯（图 3.28）。



图 3.28: 最右边的一个全加器的第一排逻辑灯可以去掉。

### 3.9.2 减法器

$a-b$  可以通过  $a+b$  的补码来得到。一个二进制数的补码是将该数每位取反，然后  $+1$ 。取反可以通过激活逻辑灯实现， $+1$  则可以利用最低位全加器的进位输入（图 3.29）。

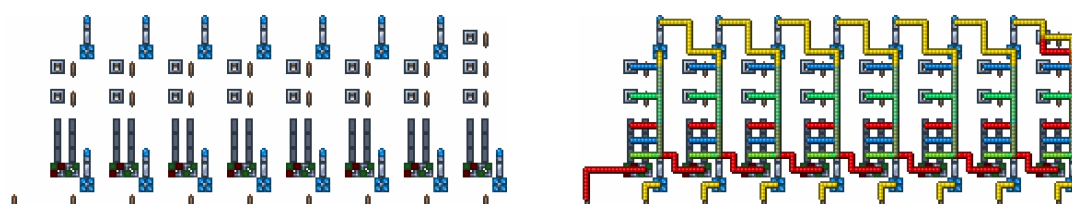


图 3.29: 右上火把亮表示做减法，灭表示做加法。黄线用来将减数取反，右上红线用来将结果  $+1$ 。

这里需要注意两点。第一点是取反的逻辑灯不能在同一个逻辑帧激活，而应该从低位到高位依次延迟一个逻辑帧激活，理由在加法器中已说明。第二点是给逻辑灯取反的同时不应当使减数的火把同时取反，也就是说给逻辑灯取反的电线与减数输入的电线不应当取同一个颜色。

### 3.10 更细致的结算顺序

在这一章开始介绍的逻辑结算顺序中，我们看到了以下的措辞：

所有激活的电源上的所有电线激活。

所有激活的电线下的所有用电器激活并响应。

对上一逻辑帧中被激活过的逻辑灯下的所有逻辑门进行逻辑判断。

这里“所有”实际上是依次。那时我们没有说“依次”是为了避免读者产生疑问。现在我们可以直接面对这个问题。

所有电路事件的结算顺序如下：



1. 在不同物理帧发生的事件，时间上先发生的事件先结算。
2. 在同一个物理帧发生的物理事件，按照程序中物理事件触发顺序进行结算。
3. 在同一个逻辑结算中不同逻辑帧发生的事件，在前面逻辑帧发生的事件先结算。
4. 当一个电源激活时会引发其上至多四种颜色的电线激活，其中红线激活先结算，然后依次是蓝线、绿线、黄线激活结算。
5. 当一个电线激活时会引发其上多个用电器激活，以激活该电线的电源为起点，对该电线覆盖的图格进行广度遍历，遍历方向顺序为下右上左，按遍历到各个用电器的先后顺序结算各个用电器的响应事件。

第1条规则很简单，时间上先发生的事件先结算，理所当然。

第2条规则就比较抽象了。什么是在同一个物理帧中发生的物理事件？什么是程序中的触发顺序？怎么知道在程序中的触发顺序？这只能通过实验或查看游戏源码推测。

第3、4、5条都是针对单个逻辑结算的。

第3条规则也很容易理解，我们之前讲到的逻辑结算机制就是基于这一条。

第4条规则说明了当一个电源激活时，先结算与该电源以红线相连的用电器，然后依次结算以蓝线、绿线、黄线相连的用电器。

第5条规则难以理解的地方可能在于“广度遍历”。这里涉及到一个算法上的问题，即泰拉瑞亚是如何寻找到一根电线上的所有用电器的？换句话说，程序是如何扫描电线经过的每个图格的？经过一些实验可以推测出扫描电线是使用的广度遍历算法。具体来说，

1. 被电源激活的图格标号为0。每次对未标号图格标的号从1开始递增。
2. 在当前标号的图格中取出与未标号电线图格相邻的标号最小的图格，依次将其下右上左的电线图格标号，标号跳过已标号的图格。
3. 重复第2步直到不存在与未标号电线图格相邻的标号图格。

在以上算法中标号顺序就是扫描电线的顺序，即用电器的结算顺序。

例如在图3.30((a))所示电路中，右击开关，则开关被标号为0。接下来依次对开关下右上左的电线图格标号为1234，然后对1号图格下右上左的电线图格标号。因为1上方的图格已经标号0，所以跳过，因此1号图格下右左的图格依次被标号567。依此类推，最后对电线所在的所有图格标号如图3.30((b))。则右击开关时火把依标号顺序关闭。

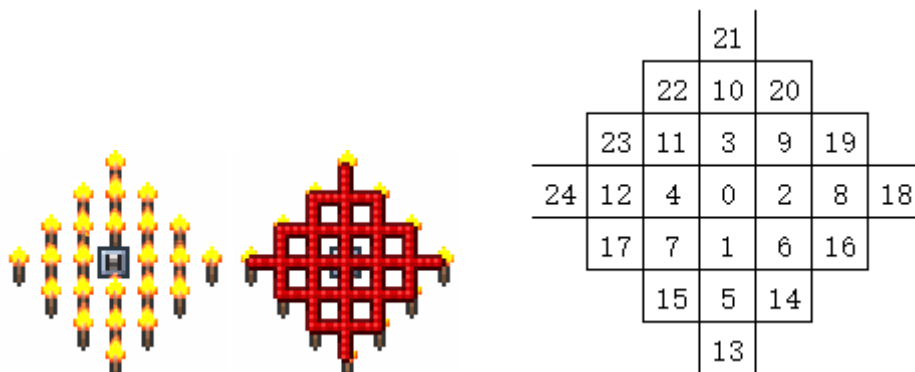


图 3.30

如果读者对于这个顺序的生成感到难以理解，不妨只记简化规则：与电源的电线距离短的先结算，电线距离长的后结算。在图 3.30 中，与电源距离为 1 的火把标号为 1, 2, 3, 4，距离为 2 的火把标号为 5, 6, 7, 8, 9, 10, 11, 12，距离为 3 的火把标号为 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24，符合这个规则。同时还可以发现，在与电源距离相等的对称位置上，下边的用电器比上边的先结算，右边的用电器比左边的先结算。

到这里读者可能有疑问，既然火把都是在同一个物理帧关闭的，如何判断出其关闭顺序呢？

### 3.10.1 实验测定两次激活的先后顺序

见图 3.31，一根电线连接开关和两个飞镖机关，右击开关时显然两个飞镖机关在同一个物理帧射出飞镖。如果在同样的距离上放两个青绿压力垫板，显然两个青绿压力垫板是在同一个物理帧激活，这就是在同一个逻辑帧发生的两个物理事件。



图 3.31

按照惯性思维，由于两个飞镖是在同一个逻辑帧射出的，那么两个青绿压力垫板也应该在同一个逻辑帧激活，那么如图 3.32 所示，第一次右击开关应当使左边火把响应。然而实际上，第一次右击开关会使右边的火把响应，这意味着两个青绿压力垫板不是在同一个逻辑帧激活的，并且左边的青绿压力垫板激活更早。



图 3.32

那么两个青绿压力垫板是否是在同一个逻辑结算中的不同逻辑帧激活的呢？看图 3.33，如果两个青绿压力垫板是在同一个逻辑结算中不同逻辑帧激活，并且左边的青绿压力垫板激活更早，那么与门会爆门。然而实际上与门并未爆门，而是激活了两次，这意味着两个青绿压力垫板并不是在同一个逻辑结算中激活的。这就是说，不同的物理电源不会参与到同一个逻辑结算中，而是等前面物理电源相关的逻辑结算完成后再执行后面物理电源的逻辑结算。

从上面的实验可以知道，左边的青绿压力垫板结算在前，右边的结算在后，这是由于激活它们的飞镖生成顺序不同。依照第 5 条规则，左边的飞镖机关的飞镖先生成，右





图 3.33: 最右边是降频电路，火把响应一次代表与门激活两次。

边的飞镖机关的飞镖后生成。从飞镖生成开始，每个逻辑帧都需要判断飞镖的碰撞，因为左边的飞镖先生成，所以判断碰撞时先判断左边飞镖的碰撞情况。因此当两个飞镖同时与青绿压力垫板碰撞时，程序将左边的碰撞事件排在前面，也就是将两个碰撞事件的触发顺序安排为左边先右边后。

我们在上面的例子中判断触发顺序是通过飞镖生成的顺序来判断青绿压力垫板的激活顺序。我们已经研究得到了青绿压力垫板的激活顺序，对于其他的物理电源的结算顺序以及不同种类物理电源之间的结算顺序，目前暂无研究，期待有心读者可以总结出更多规律。

对于结算机制了解的这么细致用处并不大，因为功能最强大的逻辑电路对这些顺序不敏感。在后续章节中才会偶尔用到这些规则。

### 3.11 思考题

#### 第 3 章 思考题

1. 做一个信号处理电路，有两个输入  $a, b$ ，要求当  $b$  在  $a$  激活后的 3 个逻辑帧内激活时输出激活。其中  $a$  在整个逻辑结算中只激活一次。
2. 做一个信号处理电路，有三个输入  $a, b, c$ ，要求当  $a, b, c$  在同一个逻辑帧激活时输出激活。其中  $a, b, c$  在整个逻辑结算中只激活一次。
3. 做一个信号处理电路，有两个输入  $a, b$ ，要求当  $a, b$  在一个逻辑结算中至少有一次在同一个逻辑帧内激活时输出激活，其中  $a$  在整个逻辑结算中至多只会在两个逻辑帧的区间内激活。
4. 证明不存在信号处理电路，有两个输入  $a, b$ ，要求当  $a, b$  在一个逻辑结算中至少有一次在同一个逻辑帧内激活时输出激活。其中  $a, b$  在逻辑结算中可能激活任意多次。

## 第 4 章 数字电路

泰拉瑞亚中有逻辑门，自然也就会有数字电路理论。又因为游戏特性，泰拉瑞亚中的数字电路体系与现实生活中的几乎完全不同。所以不要抱有以下两种幻想：

- 我是学过（数字）电路，所以我也能轻松搞定泰拉瑞亚电路。
- 我泰拉瑞亚电路很熟，所以我也一定能搞懂（数字）电路。

本章内容，一是建立泰拉瑞亚中的数字电路体系，二是利用数字电路知识进行电路的极限优化。本章专业性较强，建议本科学历以上的人阅读。

### 4.1 布尔代数

#### 4.1.1 布尔运算

一个数为**布尔值**，是指这个数的取值只能为 0 和 1。**布尔运算**是对布尔值进行的运算。我们把表达式中的布尔值与布尔常数 1 简称为**字母**。

泰拉瑞亚中一共有四种初等布尔运算，它们在电路中的意义见图 4.1：

取反  $\sim 0 = 1, \sim 1 = 0$

连接  $00 = 0, 01 = 1, 10 = 1, 11 = 0$

与  $0 \& 0 = 0, 0 \& 1 = 0, 1 \& 0 = 0, 1 \& 1 = 1$

异或  $\wedge(a_1, a_2, \dots, a_n) = \begin{cases} 1 & \text{如果 } a_1, \dots, a_n \text{ 中恰好有一个为 } 1 \\ 0 & \text{其他情况} \end{cases}$

此外，在进行分析的时候，为了便于理解，我们也使用其他一些布尔运算，它们都可以由初等布尔运算导出。

或  $a|b = \sim(\sim a \& \sim b)$

小于  $a < b = \sim a \& b$

大于  $a > b = a \& \sim b$

等于  $a == b = \sim ab$

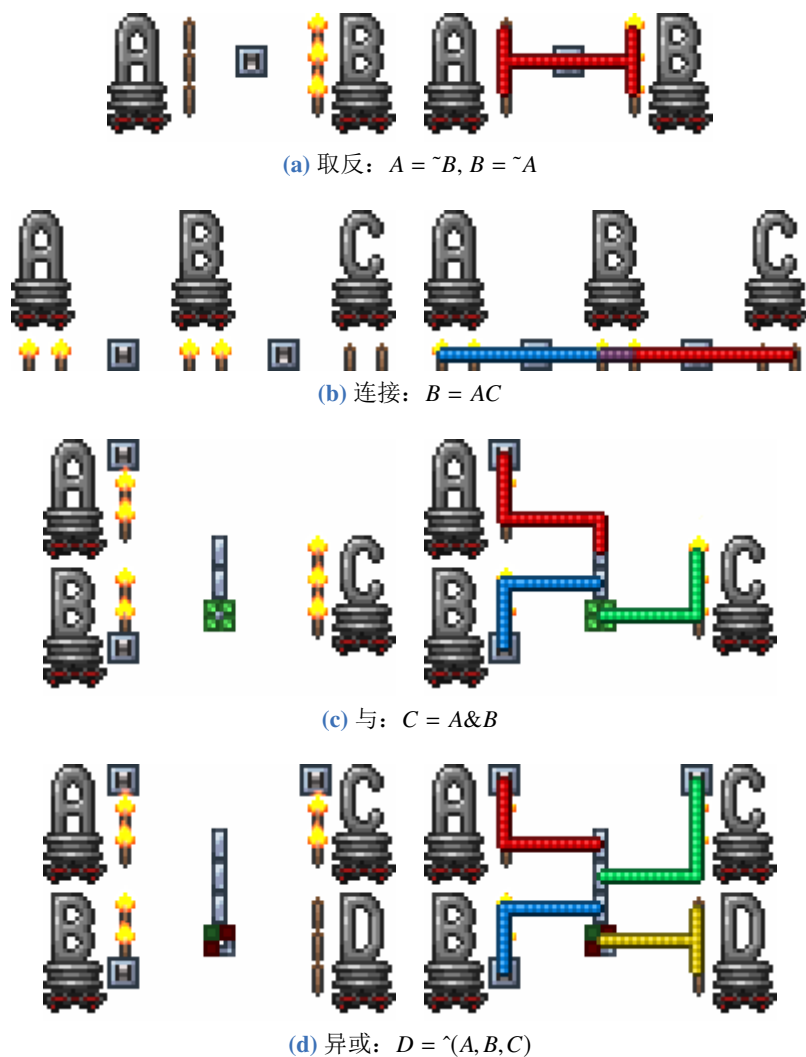


图 4.1: 初等布尔运算对应的电路。注意取反运算和连接运算不需要逻辑门。

### 4.1.2 布尔运算的性质

**交换律、分配律、结合律** 连接、与、异或都满足**交换律**, 改变这些运算下的字母顺序, 结果不变。连接、与都满足**结合律**, 但是异或不满足结合律, 因为  $1 = \sim(\sim(1, 1), 1) \neq \sim(1, 1, 1) = 0$ 。当  $a \& b = 0$  时 (这个条件必不可少!), 与和连接满足**分配律**  $ab \& c = (a \& c)(b \& c)$ 。

#### 与布尔常量的运算

连接	与	异或
$0a = a$	$0 \& a = 0$	$\sim(0, a_1, \dots, a_n) = \sim(a_1, \dots, a_n)$
$1a = \sim a$	$1 \& a = a$	$\sim(1, a_1, \dots, a_n) = \sim a_1 \& \dots \& \sim a_n$

#### 与自身的运算

- $aa = 0$
- $a \& a = a$
- $\sim a \& a = 0$

### 转化运算

- $a \& b = \sim a b \& b$
- $\wedge(a, b) = ab$
- $\wedge(a_1, a_2, a_3, \dots, a_n) = \wedge(\sim a_1 a_3 \dots a_n, \sim a_2 a_3 \dots a_n, a_3, \dots, a_n)$

**其他记号** 当我们要处理一组布尔值时，我们可以给它们编号。例如一个八位二进制数可以表示为  $[a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0]$ ，简记为  $a_{7:0}$ 。 $a_{7:0}$  的最高位是  $a_7$ ，最低位是  $a_0$ 。 $a_{7:0}$  可以分成两个四位二进制数  $a_{3:0}$ （可以叫做低四位）和  $a_{7:4}$ （可以叫做高四位）。

### 4.1.3 真值表

对于稍微复杂的逻辑表达式，例如  $\sim ab \& \sim bc \& abc$ ，我们很难一眼看出它们表达的意义。例子中的表达式只包含三种字母，它们只有 8 种取值的可能性，所以我们可以直接把这 8 种情况穷举出来，看看每种情况下这个表达式的结果是什么。

a	b	c	输出
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

表 4.1

从表 4.1 可以看出来，当且仅当  $a, b, c$  全为 1 时， $\sim ab \& \sim bc \& abc$  的值才为 1，这不就是与逻辑吗？所以我们得到  $\sim ab \& \sim bc \& abc = a \& b \& c$ ，这个逻辑看上去就舒服多了。表 4.1 就称为  $\sim ab \& \sim bc \& abc$  的**真值表**。

更多的时候，我们先得到一个真值表，然后希望得出它的逻辑表达式。例如对于显示器而言，我们在一开始只知道每个显示单元对于哪些输入应该亮，哪些输入应该暗，而不知道应该如何搭建这个逻辑。这里遇到的就是已知真值表，求解逻辑表达式的过程。本小节一开始举的那个例子里，因为真值表比较特殊，我们只通过观察就得到了它的逻辑表达式。对于简单观察得不到逻辑表达式的情况，后续一些小节会给出解决方案。

### 4.1.4 字符串的线性相关性

回顾我们把逻辑表达式中的布尔值与布尔常数 1 简称为字母。我们把由若干字母连接而成的表达式称为**字符串**，简称**串**。由 0 个字母连接成的表达式记作 0，称为**空串**。0 和 1 两个串称为**平凡串**。

注意到由连接的性质  $1a = \sim a$ ，布尔常数 1 在这个串里起到的作用是取反。例如，由 1, a, b 组成的串有 8 个<sup>1</sup>：0, 1, a, b, ab,  $\sim a$ ,  $\sim b$ ,  $\sim ab$ 。一个串中可以包含重复的字母，但是由于连

<sup>1</sup>由 n 个字母组成的串有多少个？

接的性质  $aa = 0$  和  $0a = a$ ，一对重复的字母可以直接抵消，所以我们一般忽略这种情况。

给定一组串  $s_1, \dots, s_r$ ，任选其中若干串（至少一个）做连接运算，得到的串称为这组串的一个线性组合。如果一组非平凡串有一个平凡的线性组合，那么我们称这组非平凡串线性相关。否则，如果一组非平凡串的所有线性组合都是非平凡串，那么我们称这组非平凡串线性无关。一组非平凡串的最大的线性无关子集的大小称为这组非平凡串的秩。

**例 1：**考虑三个串 **ab,abc,abcd**。令  $A = ab, B = abc, C = abcd$ ，那么  $A, B, C$  的所有线性组合为  $A = ab, B = abc, C = abcd, AB = c, AC = cd, BC = d, ABC = abd$ 。所有线性组合都是非平凡串，所以 **ab,abc,abcd** 线性无关， $\{ab,abc,abcd\}$  的最大线性无关子集就是它自己，所以 **ab,abc,abcd** 的秩是 3。

**例 2：**考虑三个串 **bc,ac,ab**。令  $A = bc, B = ac, C = ab$ ，那么  $A, B, C$  的所有线性组合为  $A = BC = bc, B = AC = ac, C = AB = ab, ABC = 0$ 。 $ABC = 0$  是平凡串，所以 **bc,ac,ab** 线性相关。 $A$  和  $B$  的所有线性组合为  $A = bc, B = ac, AB = ab$ ，都是非平凡串，所以 **bc,ac** 线性无关，所以 **bc,ac,ab** 的秩为 2。

### 4.1.5 简单逻辑及其复杂度

简单逻辑指只用一个与门或异或门就可以实现的逻辑。这个概念很好理解，但是其复杂度比我们想象的大得多。

考虑一个简单的例子。假设我们现在用一个两灯与门做出了  $a \& b$  的逻辑，在游戏中，我们要用两种不同颜色的线分别连接到两个逻辑灯。实际应用中有时出于接线考虑，可以利用与逻辑的交换律将  $a$  与  $b$  换位。但是等价的逻辑还远不止这两个。在 [小节 4.1.2](#) 的转化运算中，看到  $a \& b = \sim ab \& b$ ，同理  $a \& b = \sim ab \& a$ ，再利用交换律，又有  $a \& b = b \& \sim ab = a \& \sim ab$ 。因此， $a \& b$  的等价两灯简单逻辑一共有 6 个，不考虑交换律则有 3 个。这些等价逻辑每个都代表不同的接线方法，而多样的接线方法可以帮助我们在一些狭小空间内布线。

类似的， $a \& b \& c$  的等价三灯简单逻辑有 168 个，不考虑交换律则有 28 个。对于异或逻辑，用转化运算中的另一个公式也可以得到其等价逻辑。每一个逻辑都对应一种接线，这么多种等价逻辑就会带来很多种接线方式，也就会增加我们考虑问题的复杂度。一般情况下这么多种接线方式中总有一些可以满足要求，但是一旦不行，我们就需要引入复杂度还要高得多的单步逻辑。

168 种等价逻辑是怎么算出来的呢？如此的工作量对于人工来说，即使再有技巧都是无法接受的，更不提人工可能漏算错算的情况。对于简单逻辑，我们开发了软件 TCLC(Terraria Combinational Logic Calculator)，你只需要输入这个逻辑的真值表，它就会用穷举法找出那些满足要求的逻辑。程序下载<https://www.bbstr.net/threads/154/>。这个软件只进行了初步开发，因为目前还没有多少实际需求。如果你有更多的需求或想法，可以联系我。

### 4.1.6 单步逻辑及其复杂度

**单步逻辑**是若干简单逻辑连接而成的结果。“单步”的意思是这个逻辑在一个逻辑帧内就可以结算完毕，或者说输入与输出只间隔一个逻辑帧。类似的我们也有  $k$  步逻辑。

单步逻辑有什么用呢？如果电路中要同时执行多个简单逻辑，可以将其中若干逻辑进行连接以获取可能的简化。这个描述非常抽象，我们来看一个例子。

**全加器** 竖式计算加法时，我们实际上在从低位到高位对每一位执行同样的过程：将两加数相加，并加上低位的进位，结果为两位数，其低位写在和数上，高位向下一位进位。全加器就是执行这一逻辑的模块。

全加器接收三个输入：加数 1、加数 2、上一位进位。产生两个输出：和数、下一位进位。全加器的真值表见表 4.2，其中  $a$  和  $b$  分别表示两个加数， $c_i$  表示上一位进位， $s$  表示和数， $c_o$  表示下一位进位。

输入			输出	
$a$	$b$	$c_i$	$c_o$	$s$
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

表 4.2: 全加器真值表

我们把  $s$  的真值表输入 TCLC，得到  $s = abc_i$ ，也就是说最少可以不用任何门得到  $s$ 。把  $c_o$  的真值表输入 TCLC，发现简单逻辑至少需要四灯。但是我们如果再考虑一下  $sc_o$  的值，把它的真值表输入 TCLC 得到  $sc_o =$ ，只需要两灯就可以得到。这样一来，我们直接利用  $c_o = ssc_o$ ，把  $s$  和  $sc_o$  的输出连接起来就可以得到  $c_o$ 。这个全加器的电路图见??。

在这个例子中，我们使用了非简单的单步逻辑来处理输出  $c_o$ ，得到了更好的电路。这个优化得益于全加器需要同时产生两个输出。如果一个组合逻辑需要同时产生多个输出，就可以考虑利用连接运算。这个技巧在数显中的应用尤为突出。

??展示了一个 BCD 数显<sup>2</sup>。这个数显是现有的唯一一个可密堆叠 (??) 的 BCD 数显。可密堆叠的要求是极其苛刻的。首先，因为下方显示器的宽度相当小，控制电路的宽度也非常有限；其次，上层和下层控制单元需要可以挨在一起但是线不能连接。这些条件看上去非常不可思议，但是数显能做出来，得益于单步逻辑的超高复杂度。这个数显一共有 7 个输出，并且因为它们的秩为 7，至少需要 7 个逻辑门。输入接进来时，在逻辑灯上可以有各种接线可能；逻辑门可以选取与和异或，逻辑灯数量也可以改变；输出时，多个逻辑门可以输出到同一根线上实现连接运算；电线在显示器上也可以连接若干段。这么

<sup>2</sup>BCD 编码的概念: [https://blog.csdn.net/qj\\_33750826/article/details/53004685](https://blog.csdn.net/qj_33750826/article/details/53004685)

算下来，不考虑正确性的情况下，一个简单数显的接线排列组合可以有成千上万种。从这么复杂的情况中找出一种可行方案不算太困难，但是也需要相当的观察力与经验。

类似的例子还有使用加三移位法做的二进制与十进制互化的逻辑单元，相关内容会在后面讲到。

## 4.2 组合逻辑与时序逻辑

泰拉瑞亚中一共有三种功能不同的逻辑门：与门、异或门和故障逻辑门。电路中的信号传递方式也分为两种：状态传递和激活传递。状态传递是指，如果在一根线上放一个火把，那么火把亮表示这根线的状态是 1，火把灭表示这根线的状态是 0；激活传递是指，在某个特定的逻辑帧，电线激活表示 1，不激活表示 0。所有电线全部为状态传递的电路称为组合逻辑电路，有至少一根电线为激活传递的电路称为时序逻辑电路。

组合逻辑电路的特点是电路中只有与门和异或门，并且每根线只在某个固定的逻辑帧激活（逻辑延迟器中的爆门除外）。组合逻辑的每个输出都对应一个真值表。

时序逻辑电路利用激活与否来传递信息，同一根电线可能在不同的逻辑帧激活，并且不同逻辑帧激活可能表示不同的信号。递次电路就是一种典型的纯时序逻辑电路。

与现实的数电不同，泰拉瑞亚中的组合逻辑和时序逻辑没有功能上的明确限制。有很多电路既可以用组合逻辑实现，也可以用时序逻辑实现。组合逻辑电路和时序逻辑电路各有优劣，一个电路选择使用组合还是时序，取决于电路本身的特点。

- 在组合逻辑电路中，整个电路的运行状态可以直接通过读灯得到，如果出了 bug，直接看哪个灯的状态不对，就能找到问题。相比较而言，如果时序逻辑电路出了 bug，除非借助 MechScope，否则很难进行调试。
- 由于逻辑灯接线取向的问题，时序逻辑电路中经常要出现违背逻辑灯取向的反向绕线，接线比组合逻辑更困难。
- 为了逻辑同步，复杂的电路中经常会用到逻辑延迟器。组合逻辑电路中每个数据都需要设计一个独立的逻辑延迟器，而时序逻辑电路中可以让所有数据共享一个逻辑延迟器。将所有待传输的数据暂时存储起来，然后由中央逻辑延迟器发送信号释放这个数据。这一点现在很难看明白，后面在除法器部分会体现的很清楚。

很多电路概念都分为状态和激活两种情况，也可以分别归类为组合逻辑和时序逻辑。例如加重压力板属于组合逻辑电源，而普通压力板属于时序逻辑电源；火把属于组合逻辑用电器，而像素盒属于时序逻辑用电器。

## 4.3 计数系统

计算机中一律使用二进制进行计算，而且计算机中存储数字的一般形式是固定数位的非负整数，例如 8 位二进制数的取值就是 0~255，16 位二进制取值是 0~65535。那么计算机中如何表示负数和小数呢？这就是这一节中将要介绍的内容。



### 4.3.1 无符号整数、有符号整数

$n$  个二进制位，只用来表示非负整数的话，可以表示  $2^n$  个数，取值范围为  $0 \sim 2^n - 1$ 。如果还要用来表示有符号整数，那么需要拿出最高位来做符号位：最高位为 1 代表这个数为负数。例如 00000001 表示 1，10000001 表示 -1。简单来说，把  $a$  变成  $-a$ ，只需要把最高位取反。这样一来  $n$  位有符号整数的取值范围为  $-2^{n-1} + 1 \sim 2^{n-1} - 1$ 。

实际应用中，我们使用补码表示有符号整数。补码的规则初看起来很奇怪：把  $a$  变成  $-a$ ，需要把  $a$  的所有数位取反，然后 +1。例如 1 的二进制表示为 00000001，要得到 -1 的补码，就要把 1 的所有数位取反得到 11111110，然后加 1 得到 11111111，11111111 就是 -1 的补码表示。反过来，把 11111111 的所有数位取反得到 00000000，然后加 1 得到 00000001，恰好是 1 的补码表示。 $n$  位二进制补码的取值范围为  $-2^{n-1} \sim 2^{n-1} - 1$ 。

为什么要这么做？这么做有什么好处？要解释这些问题，我们回到更熟悉的十进制，假设我们处理的是 8 位十进制整数。十进制补码是这么定义的： $-a = 10^8 - a$ 。例如，-1 的补码是 99999999， $1 + 99999999 = 100000000$ ；-12345 的补码是 99987655， $12345 + 99987655 = 100000000$ 。一般的， $-a + a = 100000000$ ，而 100000000 在 8 位十进制整数系统中，其最高位溢出，被丢弃，只保留最低的 8 位，即 00000000，这样就自然满足了  $-a + a = 0$ 。利用补码，我们不再需要在计算加减法的时候对正负进行判断，直接加就完事了。

回到 8 位二进制，如何做有符号的加减法？

#### 例 1：计算 $-3 + 5$

3 的补码是 00000011，-3 的补码为  $11111100 + 1 = 11111101$ 。5 的补码是 00000101。计算  $-3 + 5$ ，直接把它们补码相加： $11111101 + 00000101 = 100000010$ ，最高位溢出，结果为 00000010，答案是 2。

#### 例 2：计算 $123 - 12$

123 的补码是 01111011。12 的补码是 00001100，-12 的补码是  $11110011 + 1 = 11110100$ 。直接把 123 和 -12 的补码相加： $01111011 + 11110100 = 101101111$ ，最高位溢出，结果是 01101111，答案是 111。

### 4.3.2 固定数位小数

1

### 4.3.3 浮点数

1

## 4.4 逻辑电路的运算速度

一些简单的研究表明，一个电路的复杂度主要取决于被激活的电线的格次。所以在电路较复杂时，想办法减少电线长度，或者减少一根电线的激活次数就可以减少电路的复杂度。

如果电路的复杂度过高，会导致主机 CPU 速度不足，从而限制物理帧率。一个简单的例子是使用电线连接四万个火把（ $200 \times 200$ ），然后使用满频驱动激活这些火把。在笔者的电脑上，打开驱动后，可以看到物理帧率降到了？；打开五秒计时器，发现其周期变成了？。这个速度会受到电脑性能的影响。

### 4.4.1 多级递次电路

假设我们要做一个超大规模的递次电路：1000-递次电路。如果我们直接摆开 1000 个逻辑门，这样的递次电路激活一次，就会激活连接所有顶灯的线（约 2000 格）和对应逻辑门上的线（4 格），我们把逻辑门上的线忽略。那么递次电路运行一个周期，电线激活的格次就大约是  $2000 \times 1000 = 2000000$ 。

如果我们把递次电路平分成两段，分别记为  $A$  和  $B$ ，一开始使用  $A$ ，等  $A$  运行完一个周期以后切换到  $B$ ，这个切换可以通过一对故障逻辑门完成。我们把这一对故障逻辑门叫做一级递次， $A$  和  $B$  叫做二级递次。那么整个电路运行一个周期，一级递次激活了 1000 次，格次为数千；二级递次各激活了 500 次，格次约为  $1000 \times 500 \times 2 = 1000000$ 。这样一来，总的复杂度就减少了一半。如果继续增加级数，复杂度还会继续减少。

传统递次电路周期复杂度为  $O(n^2)$ ，平均单次激活复杂度为  $O(n)$ 。使用多级递次电路，周期复杂度最少为  $O(n \log n)$ ，平均单次激活复杂度最少为  $O(\log n)$ 。

超大规模递次电路可用于传送器定格动画，一个实例为<https://www.bilibili.com/video/av46694445>

## 4.5 算术电路

### 4.5.1 加法器（Adder）

加法器的输入是两个整数，输出是两个输入之和。加法器的解决方案有很多，一个最直接的方法就是串联全加器。

全加器是什么？我们来看二进制加法的竖式计算。计算  $a_{7:0} + b_{7:0}$  时，首先列出图 4.2 所示的竖式并且把两个加数填入。然后计算  $a_0 + b_0$ ，这个结果可能是 00, 01, 10。结果的低位直接就是和的最低位，写在  $s_0$  的位置。结果的高位是最低位向前的进位，写在  $c_1$  的位置。然后计算  $a_1 + b_1 + c_1$ ，把结果的低位填入  $s_1$ ，高位作为进位填入  $c_2$ 。依此类推，在每一位上计算  $a + b + c$ ，把结果的低位填入  $s$ ，高位填入下一位的进位。这就是全加器的工作。

全加器接收三个输入  $a, b, c_i$ ，其中  $a, b$  分别是两个加数的对应数位， $c_i$  是低位产生的进位。全加器产生两个输出  $s, c_o$ ，其中  $s$  是和的对应数位， $c_o$  是向高位的进位。全加器

加数 1	$a_7$	$a_6$	$a_5$	$a_4$	$a_3$	$a_2$	$a_1$	$a_0$	
加数 2	$b_7$	$b_6$	$b_5$	$b_4$	$b_3$	$b_2$	$b_1$	$b_0$	
进位	$c_7$	$c_6$	$c_5$	$c_4$	$c_3$	$c_2$	$c_1$	$c_0$	
和	$s_8$	$s_7$	$s_6$	$s_5$	$s_4$	$s_3$	$s_2$	$s_1$	$s_0$

图 4.2

的真值表如表 4.3。

$a$	$b$	$c_i$	$s$	$c_o$	$sc_o$
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	1
0	1	1	1	0	1
1	0	0	0	1	1
1	0	1	1	0	1
1	1	0	1	0	1
1	1	1	1	1	0

表 4.3: 全加器的真值表

在面临多输出的单步逻辑的时候，最直接的思路就是分别考虑各输出的逻辑。这里可以得到  $s = \sim(\sim a, \sim b, \sim c, abc)$ ， $c_o = abc$ 。 $c_o$  只需要用一个灯，而  $s$  需要用四个灯。如果觉得这个结果不好，还可以考虑各输出的线性组合的逻辑。这里  $c_o$  和  $s$  除了它们本身以外只有一个线性组合  $sc_o$ ，它的真值表已经列在了表 4.3 中。得到  $sc_o = \sim(ac \& \sim bc)$ ，这个逻辑只需要用两个灯。最终的输出  $s = sc_o c_o$ 。电路如 ?? 所示。

有了全加器，直接把八个全加器首尾相连，就得到了一个加法器 (??)。这个加法器要求两个加数从低位到高位依次延迟一个逻辑帧输入。

除了用组合逻辑，我们还可以用时序逻辑做加法器。降频电路有逢二进一的特性，所以可以直接用降频电路做一个加法器 (??)。与组合逻辑加法器不同，时序逻辑加法器实质上是累加器，如果不复位，那么每次输入都会直接累加到上一次的结果上。

在加法器上，时序逻辑的另一个优点是，把降频电路改成递次电路，就可以直接做任何进制的累加。中的进制转换实际上就是做了一个十进制累加器，每修改一个二进制数位就向累加器中加一个值或减一个值。

### 4.5.2 减法器 (Subtractor)

有了补码这个工具，减法器 and 加法器就可以共用一个电路，我们只需要增加一些部件将其中一个加数取反加一。取反的操作是容易的，而加一的操作正好可以借用 ?? 中我们没有用到的  $c_0$ 。做出的组合逻辑加减法器如 ?? 所示，需要注意的是加法器的输入从低位到高位延迟一个逻辑帧，所以对每个数位取反的时候也要从低位到高位延迟一个逻辑帧。

时序逻辑的减法器同样使用补码运算，这里不详细给出具体实现。

### 4.5.3 比较器 (Comparator)

我们在??中已经给出了比较器的一种解法。这里我们给出比较器的另一种解法。比较两个数的大小，除了逐位比较之外，还可以直接将两个数做减法，并判断结果的正负性。减法器我们已经做过了，而补码表示中，一个数的正负可以直接通过最高位得出。这里有两个细节的问题。无符号整数没有符号位，如何比较两个无符号整数的大小？有符号整数的两数之差可以达到  $127 - (-128) = 255 = 11111111$ ，而这个数作为有符号整数实际上表示  $-1$ ，即小于  $0$ 。如何处理这些特殊情况？事实上我们这里的比较器只能比较无符号整数，而所谓的“最高位”实际上指的是图 4.2 中的溢出位  $s_8$ 。对于有符号整数有两种处理方法：第一种是首先判断符号，如果符号相等，再做无符号整数的比较；第二种是把两个数都加  $128$ ，这样就把  $-128 \sim 127$  的范围变成了  $0 \sim 255$ ，然后做无符号整数的比较即可。

比较两个数是否相等，也有两种解法，一种是直接逐位比较，另一种是判断它们的差是否为  $0$ 。这两种方法并没有绝对的优劣，因为它们往往是附加在其他电路上的。

### 4.5.4 乘法器 (Multiplier)

乘法是加法的推广，乘法器也是加法器的推广。我们首先来看二进制乘法的竖式计算。因为乘法规模较大，这里给出 4 位乘法的例子（图 4.3）。

				$a_3$	$a_2$	$a_1$	$a_0$
				$b_3$	$b_2$	$b_1$	$b_0$
				$b_0 \& a_3$	$b_0 \& a_2$	$b_0 \& a_1$	$b_0 \& a_0$
			$b_1 \& a_3$	$b_1 \& a_2$	$b_1 \& a_1$	$b_1 \& a_0$	
		$b_2 \& a_3$	$b_2 \& a_2$	$b_2 \& a_1$	$b_2 \& a_0$		
	$b_3 \& a_3$	$b_3 \& a_2$	$b_3 \& a_1$	$b_3 \& a_0$			
$h_3$	$h_2$	$h_1$	$h_0$	$l_3$	$l_2$	$l_1$	$l_0$

图 4.3: 乘法竖式计算

从图 4.3 可以看出来，做一个 4 位乘法实际上就是做 4 个数的加法。两个数的加法我们会了，四个数怎么加？这里又有两种选择，一种是直接重新设计一个四个加数的加法器，另一种是把两个数的加法器串联起来。

### 4.5.5 除法器 (Divider)

1

### 4.5.6 移位器 (Shifter)

1

### 4.5.7 随机数生成器 (Random Number Generator)

1

### 4.5.8 进制转换

这里主要讲二进制与十进制互化的加三移位法。其他进制转换可以用类似的思路。

对于一个二进制数  $a_{n:0}$ ，如何手算得到它的十进制？

**方法一** 直接用定义式  $a_{n:0} = a_n 2^n + a_{n-1} 2^{n-1} + \cdots + a_0 2^0$ 。要利用这个式子将二进制转换为十进制，需要做一个十进制加法器，然后存储 2 的各次幂的十进制值。<https://www.bilibili.com/video/av40474377/>就是用的这种方法。

**方法二** 把定义式改写成  $a_{n:0} = 2(\cdots(2(2(2a_n + a_{n-1}) + a_{n-2}) + a_{n-3}) + \cdots) + a_0$ 。把这个过程分步来看，就是：

- 1 取出二进制数的第一位  $a_n$  作为结果。（当前结果为  $a_n$ ）
- 2 把上一步的结果乘 2，然后加上下一位  $a_{n-1}$ ，作为结果。（当前结果为  $2a_n + a_{n-1} = a_{n:n-1}$ ）
- 3 把上一步的结果乘 2，然后加上下一位  $a_{n-2}$ ，作为结果。（当前结果为  $2(2a_n + a_{n-1}) + a_{n-2} = a_{n:n-2}$ ）
- $\vdots$
- $n+1$  把上一步的结果乘 2，然后加上下一位  $a_0$ ，作为结果。（当前结果为  $2(\cdots(2(2(2a_n + a_{n-1}) + a_{n-2}) + a_{n-3}) + \cdots) + a_0 = a_{n:0}$ ）

这实际上是在反复循环一个过程：乘 2，然后加上 1 或 0。接下来考虑十进制如何进行这个计算。

这里我们选用 BCD 码中的 8421 码来表示十进制，即每个十进制位用 4 位二进制表示，这 4 位二进制占的大小分别为 8,4,2,1。一个十进制数乘 2，相当于每一位都乘 2，然后补上对应的进位（乘 2 的时候不会产生越位进位，即个位乘 2 的进位不会影响到百位）。这个过程写成算法形式就是：

1. 该位乘 2。
2. 如果达到了 10，那么减 10 并向高位进位。
3. 加上下一位进位。

继续观察，执行完前两步后，这一位上一定是偶数，也就是说 BCD 码的最低位是 0。那么第三步的加上低位进位其实就是把低位进位放到了 BCD 码的低位上。再换种说法，把十进制数乘 2 后，每一位上 BCD 码的最低位就是低位的进位，就是乘 2 之前该位上数字的两倍。

## 4.6 存储电路

1

### 4.6.1 多路选择器

1

#### 4.6.2 只读存储器 (Read-Only Memory)

1

#### 4.6.3 只写存储器 (Write-Only Memory)

1

#### 4.6.4 随机存储器 (Random Access Memory)

1

#### 4.6.5 寄存器 (Register)

1

#### 4.6.6 栈 (Stack)

1

### 4.7 分段显示器化简理论

1

#### 4.7.1 分段显示器主要结构

1

#### 4.7.2 显示矩阵、分段矩阵和数字矩阵

1

#### 4.7.3 矩阵的初等变换

1

#### 4.7.4 化简原理与细节

1

### 4.8 处理器结构

1



### 4.8.1 汇编语言

1

### 4.8.2 机器语言

1

### 4.8.3 处理器的拓扑结构

1

### 4.8.4 数据路径 (Data Path)

1





## 第 5 章 电路文档

---

在前面章节中介绍电路的目的是帮助读者理解电路原理，所以有些电路仅给出了最简单的例子。在本章中，会直接给出一些电路的结果供参考。对于较难的电路会给出思路。



图 5.1: 区域重生感应器。在玩家进入世界或玩家复活后，首次进入傀儡附近时激活，有小延迟。

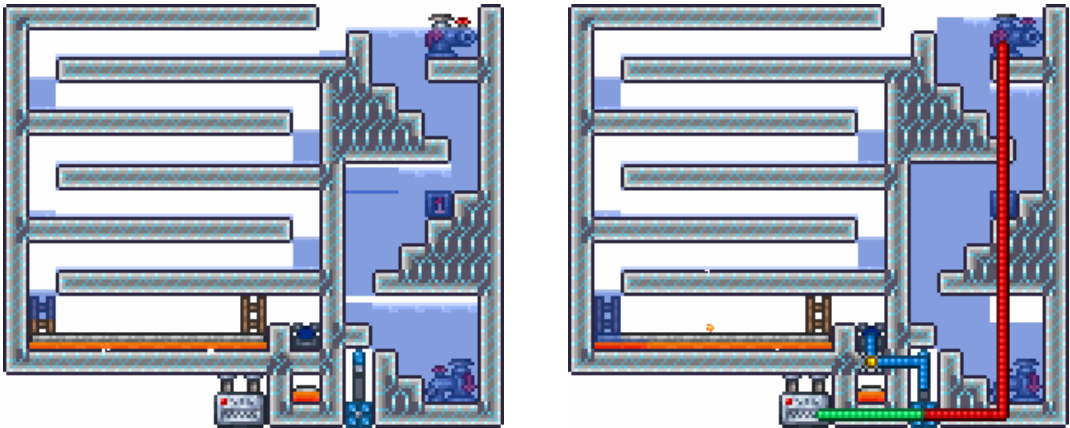


图 5.2: 开服感应器。在玩家进入世界时激活。

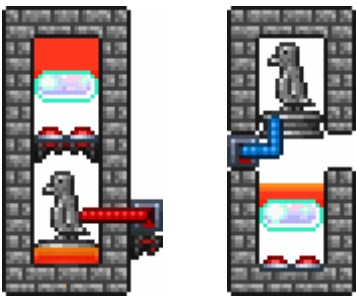


图 5.3: 血月感应器。打开一秒计时器后，压力板会不断激活。



图 5.4: 血月 & 雨天感应器。打开一秒计时器后，血月时右边火把不断激活，雨天且不是血月时左边火把不断激活。

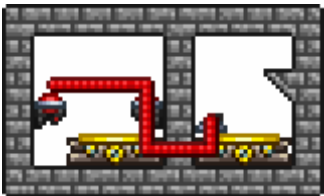


图 5.5: 击退感应器。玩家被击退时传送走。用于检测穿墙怪。



图 5.6: 传统递次



图 5.7: 斜式传统递次



图 5.8: 密排传统递次

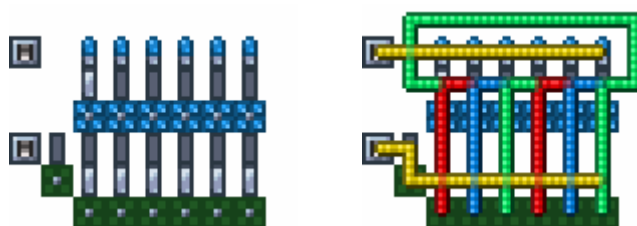


图 5.9: 密排带复位传统递次



图 5.10: 带复位的传统递次



图 5.11: 传统双向递次

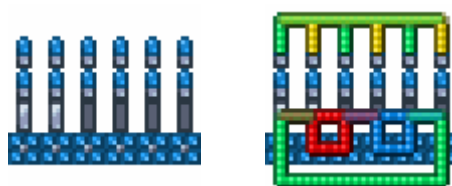


图 5.12: 密排传统双向递次



图 5.13: 带复位的传统双向递次

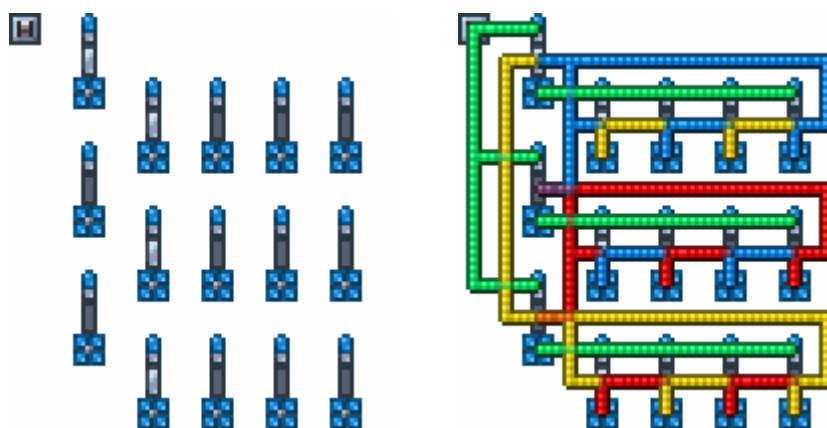


图 5.14: 周期为 12 的二级递次

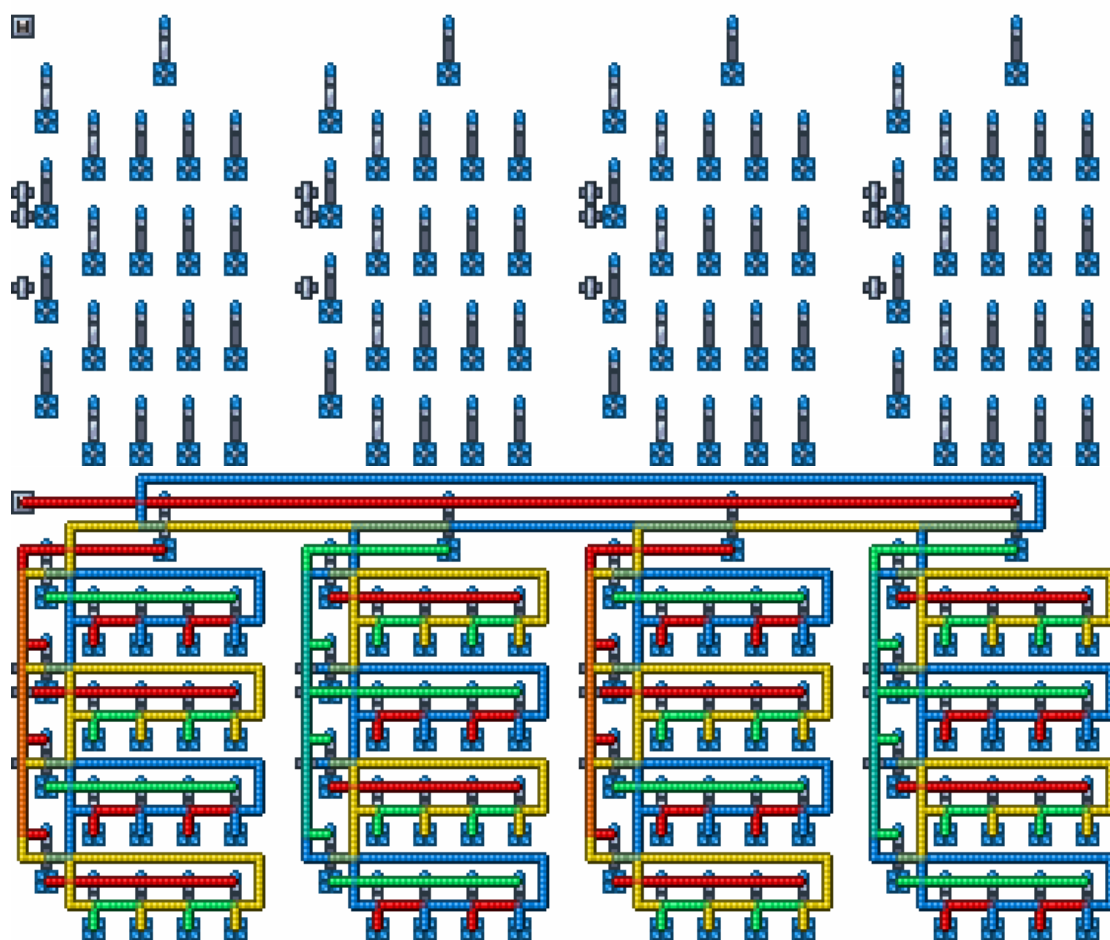


图 5.15: 周期为 64 的三级递次



图 5.16: 周期为 15 的推广递次



图 5.17: 带复位的推广递次

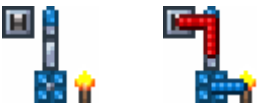


图 5.18: 降频 2



图 5.19: 降频 3



图 5.20: 降频 4



图 5.21: 降频 1-2

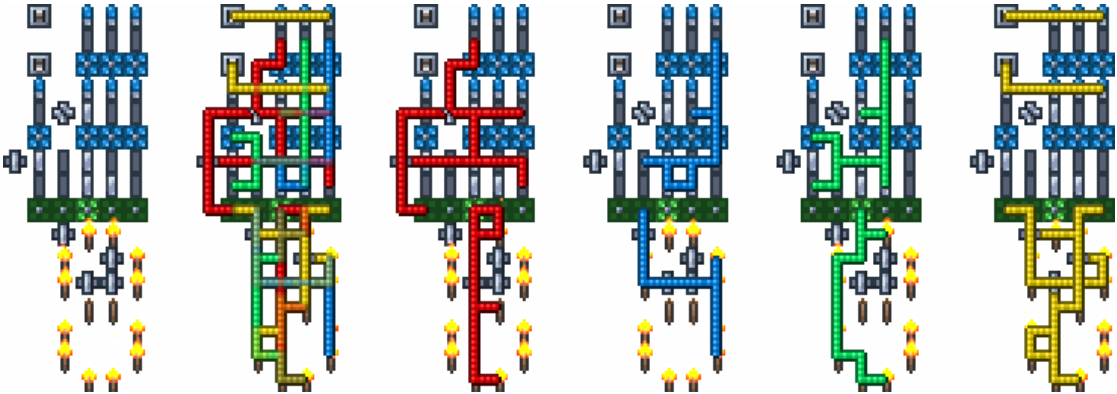


图 5.22: 带复位六进制计数器

## 5.1 传感器

## 5.2 递次电路

### 5.2.1 传统递次电路

### 5.2.2 传统双向递次电路

### 5.2.3 多级递次电路

### 5.2.4 推广递次电路

## 5.3 降频电路

### 5.3.1 固定数值的降频

### 5.3.2 交错数值的降频

### 5.3.3 可变数值的降频

## 5.4 数字显示

### 5.4.1 BCD 数显

### 5.4.2 十进制计数器

### 5.4.3 六进制计数器

### 5.4.4 月相显示

## 5.5 随机电路

### 5.5.1 随机多选一

### 5.5.2 随机分两组

### 5.5.3 随机分多组

## 5.6 操纵板

高频三向操纵板

低频四向操纵板

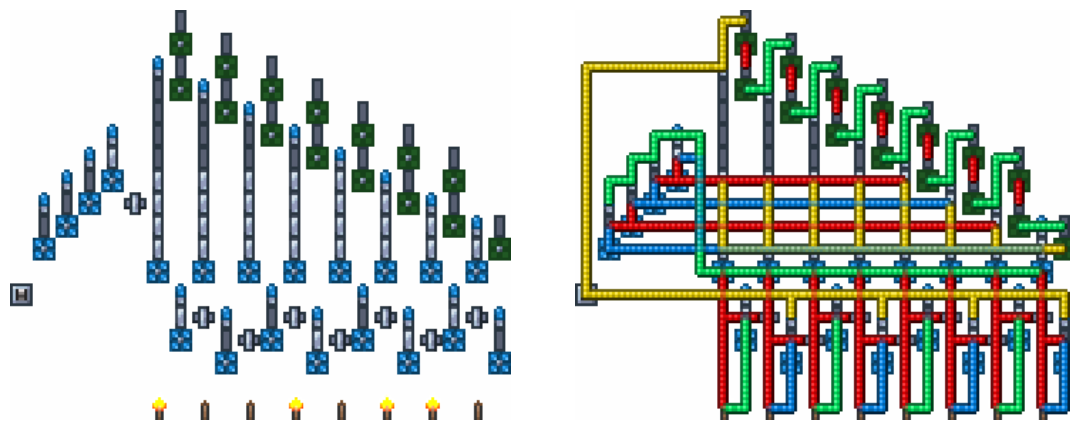


图 5.23: 随机八选四

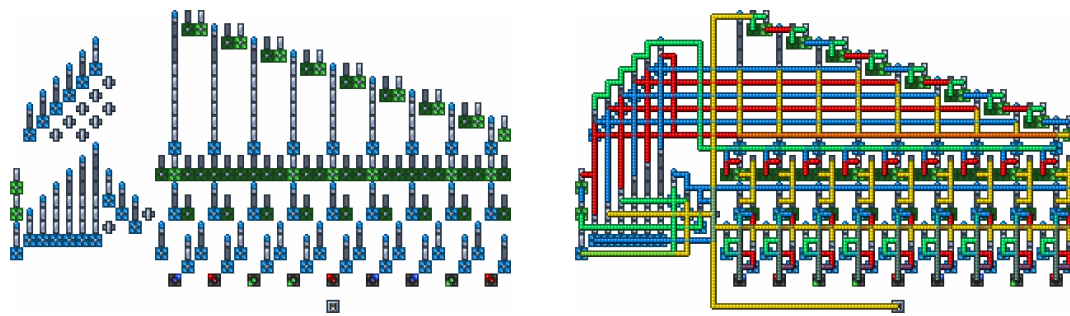


图 5.24: 九个输出随机分成 3+3+3, 占地最小

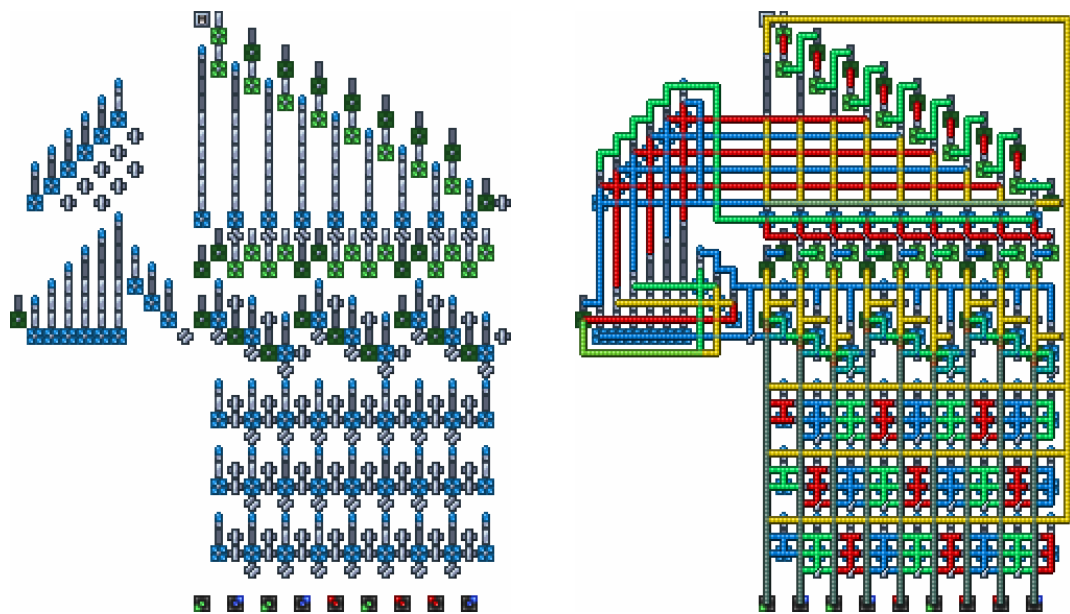


图 5.25: 九个输出随机分成 3+3+3, 最窄



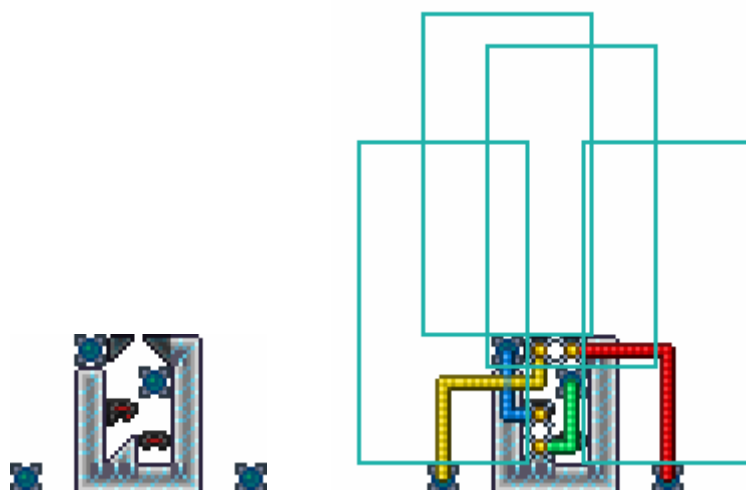


图 5.26: 低频四向操纵板（设计 1）

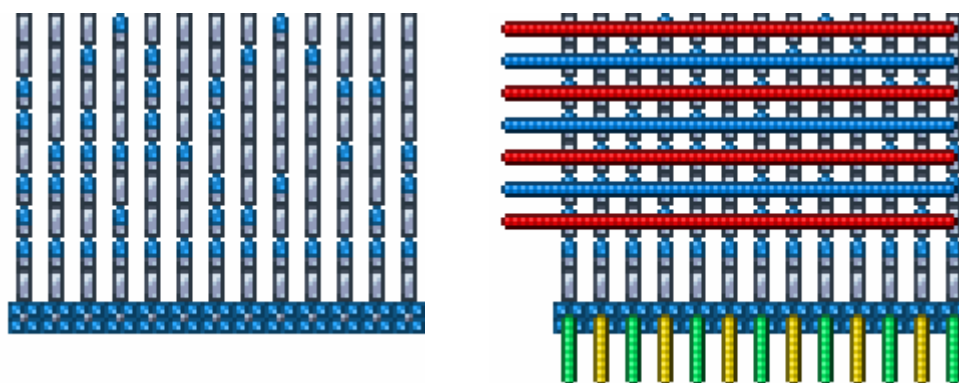


图 5.27: ROM 设计 1，横向输入纵向输出

## 5.7 像素盒显示器

## 5.8 矩阵显示器

## 5.9 算术电路

### 5.9.1 加减乘除

### 5.9.2 状态位运算

### 5.9.3 激活位运算

### 5.9.4 移位

## 5.10 存储器

RAM 设计 123 栈

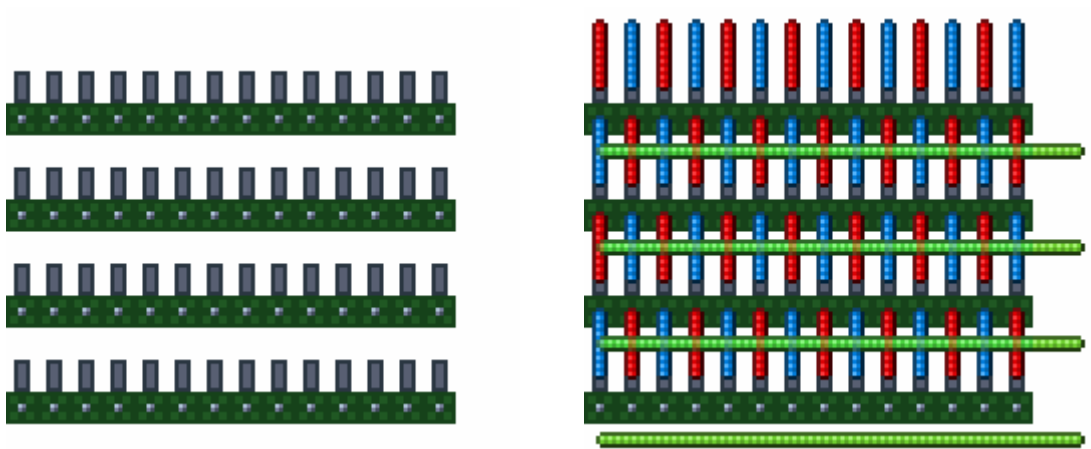


图 5.28: ROM 设计 2, 纵向输入横向输出

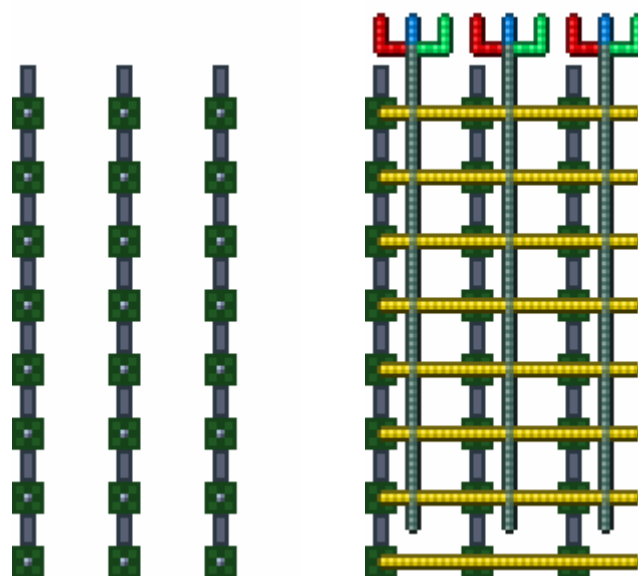


图 5.29: ROM 设计 3, 纵向输入横向输出

## 第 6 章 更多电路专题

在前面我们已经学习了所有电路原理。在这一章中我们以电路功能分类，介绍一些常用的电路模块。当然在这之前，我们需要详细地了解泰拉瑞亚中所有电路物品的性质，这样才能知道我们可以做什么。泰拉瑞亚 wiki 中当然包含每个电路物品的词条，所以这里仅对 wiki 中没有的信息进行补充。

### 6.1 驱动与延时器

驱动与延时器功能和原理都类似，只不过驱动是重复输出，延时器只有一次输出，所以放到一起。

#### 6.1.1 降频技术

灵活使用递次电路和降频电路可以将已有的驱动时长增加为任意整数倍（图 6.1）。

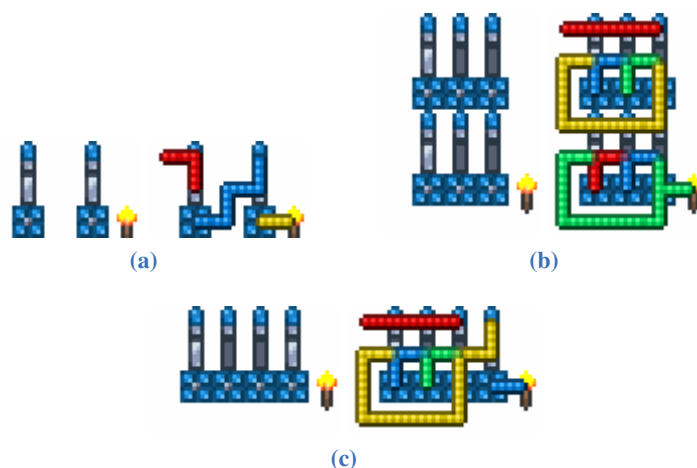


图 6.1: ((a))两个降频电路连接，红线每激活 4 次火把响应一次；((b))两个递次电路连接，红线每激活 9 次火把响应一次；((c))降频电路与递次电路连接，红线每激活 6 次火把响应一次。

使用上面的方法，当需要获得较大的质数倍（例如 23 倍）时间时使用递次电路体积过大，此时可以利用故障逻辑门的控制功能灵活地将多个降频的驱动结合（图 6.2）。

#### 6.1.2 计时器串联技术

将不同时间的计时器连接起来可以得到它们时间之和（图 6.3）。将相同时间的计时器连接在一起如何呢？

wiki 告诉我们，当相同时间的计时器 A 与计时器 B 连接时，如果 A 激活将 B 打开，那么一个计时周期过后，B 比 A 先结算，因此 B 激活将 A 关闭，A 关闭从而不会激活。

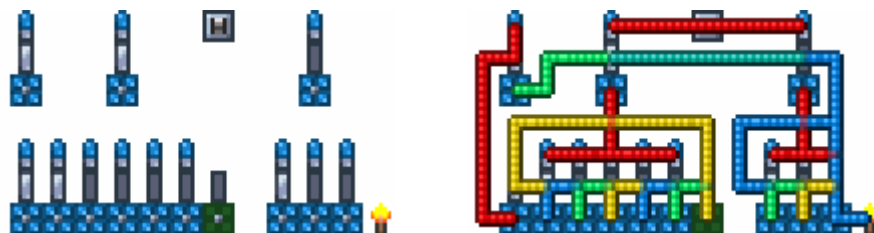


图 6.2: 开关每激活 23 次火把响应一次。上面的右边两个故障逻辑门用来控制，左边的输出接到计数为 20 的模块（5-递次电路与两个降频电路连接），右边的输出接到 3-递次电路。开关激活时，起初左边输出，右边不输出，左边计数。当左边计数到 20 时上面的绿线激活，改变控制用的有效逻辑灯，改为右边输出，左边不输出，右边计数。当右边计数到 3 时激活火把并将控制用的有效逻辑灯改回。



图 6.3: 红线激活后 8 秒火把激活。红线激活时打开 5 秒计时器；经过 5 秒，5 秒计时器激活，打开 3 秒计时器；经过 3 秒，3 秒计时器激活，关闭 5 秒计时器，激活火把，并通过换线器将自己关闭。

因此相同时间的计时器连接在一起也可以得到它们的时间之和（图 6.4）。灵活结合这个技术与降频技术就可以得到占用体积相当小的整数秒计时器。



图 6.4: 左边红线激活后 5 秒火把激活。红线激活时打开左一 1 秒计时器；经过 1 秒，左一 1 秒计时器激活，打开左二 1 秒计时器；经过 1 秒，左二 1 秒计时器（在左一 1 秒计时器之前）激活，关闭左一 1 秒计时器，打开左三 1 秒计时器；依此类推。右一 1 秒计时器激活时关闭右二 1 秒计时器，激活火把，并通过换线器将自己关闭。

### 6.1.3 自定义半砖驱动

使用假人驱动可以得到 60Hz 的驱动。如果要进行降频，除了使用降频技术以外，还可以修改半砖装置（图 6.5）。



图 6.5: ((a))只使用一个压力板，频率降为 30Hz；((a))加长半砖，频率降为 20Hz。

### 6.1.4 随意操纵时间

我们已经可以利用低频驱动得到任意秒的长时间，使用高频驱动得到任意帧的短时间。将这些装置结合起来就可以得到任意长度的时间。<https://www.bilibili.com/video/av13658918>中秒杀拜月教邪教徒就是通过计时器 + 飞镖机关精确控制时间完成的。

### 6.1.5 其他驱动摘要

由于使用计时器和假人驱动已经可以随意控制时间，其他驱动也就逐步淡出。但是为了保留思路，仍介绍这些驱动，说不定什么时候有奇效。

- 生物驱动：利用生物行走速度固定构造的驱动。往往利用雕像刷怪。与玩家距离不能太远，否则生物会消失。雕像怪物速度测试<https://www.bilibili.com/video/av22739934>。在 1.3.0.1 版本引入傀儡之前，半砖驱动一般使用骷髅雕像生成的骷髅。
- 柱形驱动：利用下层叠平台的稳定间隔构造的驱动。链接<https://www.bilibili.com/video/av23028215>。
- 传送带驱动：已介绍。虽然仍为满频驱动，但是需要限制玩家自由。
- 机关驱动：利用各种可以发射射弹的电路物品与青绿压力垫板构造的驱动。由于射弹速度各不相同，并且还受液体减速影响，控制不如假人驱动简单。但是由于占用空间可以很小，该类驱动仍有一定价值。
- 传送机驱动：利用传送机传送对象相对于传送区域位置不变的特性。见图 6.6

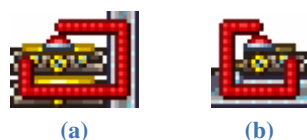


图 6.6: ((a))踩踏红压力板时触发传送机，传送后会悬空一格，坠落到压力板上时继续传送，如此循环；((b))传送后悬空半格。

## 6.2 传感器

泰拉瑞亚中已经自带了很多传感器，例如压力板、感应器等。在实际使用中，我们有时需要检测游戏自带传感器检测不了的东西，那么就需要另外设计传感器。

### 6.2.1 开服感应器

所谓开服感应器，就是在打开地图时激活的电源。只要地图不关闭，该电源就不会再次激活。

最容易想到的就是在重生点放上玩家感应器，并把玩家传送走。但是这样一来，在玩家回程时会再次激活。使用床传送技术<sup>1</sup>可以避免这一点，但是无法在退出地图时自动

<sup>1</sup>[https://v.youku.com/v\\_show/id\\_XMTgONzYxNDg00A](https://v.youku.com/v_show/id_XMTgONzYxNDg00A)

重置。

从打开地图开始，玩家第一次接近傀儡时傀儡影子会在傀儡上生成。傀儡是家具而傀儡影子是敌怪，它们分开结算：家具是固定的，而影子可以移动、受伤害、受 debuff。虽然影子不会主动移动，但是会被动移动，例如坠落、传送机传送、半砖传送。当影子受伤害时家具播放动画。影子与家具所在格由于各自的原因均不能摆放前景物块<sup>2</sup>。

只要傀儡影子在重生点附近，那么傀儡影子在打开地图时就会重新生成在傀儡上。

最简单的开服感应器如图 6.7。傀儡悬空，下面放有压力板。每次打开地图后傀儡影子生成，掉落在压力板上激活压力板。这个开服感应器略有延迟，延迟长度是傀儡影子掉落到压力板上的时间。一般来说这么短的延迟不会有什么问题。



图 6.7

如果想要更短的延迟，可以考虑在开图瞬间用传送机将傀儡影子传送走，并做一些处理使传送机再次激活时不会将影子传送回（图 6.8）。

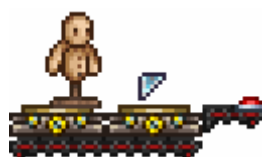


图 6.8

上面的装置中，传送机激活后 1 帧，傀儡影子就被半砖推离传送机并触发红压力板，从而不会再传回。但是玩家感应器和加重压力板不会触发传送机，玩家出生在重生点处也不会触发普通压力板。只能用玩家感应器或加重压力板触发机关，然后用青绿压力垫板激活传送机，这样一来在触发机关到青绿压力垫板激活之间又有一个短延迟。

利用液体可以实现无延迟的开服感应器。打开地图的等待界面中有一项是“正在摆放液体”。这一步是将地图中所有不稳定的液体转移到最低处。在图 2.38 中，刷水机不断地生成水，水进入到左边的细长通道，并被下面的熔岩轨道吸收。当通道足够长并且刷水速度足够快时，通道中始终有水存在。此时退出地图并重新加载地图时，通道中的水被自动放置到最低处的液体感应器上，从而液体感应器激活。其余部分的功能是：将液体感应器上的水排走；打开刷水的 1 秒计时器。这个装置的触发是没有延迟的，但是会在游戏中一直运行刷水机，可能影响电脑性能。

## 6.2.2 方向感应器

方向感应器被广泛地使用在电路游戏中，它可以检测到玩家的上下左右移动操作。方向感应器的一种方案如图 6.9 所示。另一种方案请参考<https://www.bilibili.com/vid>

<sup>2</sup>前景物块不能摆放在家具图格上或生物碰撞箱上。

eo/av23633364/?p=7。

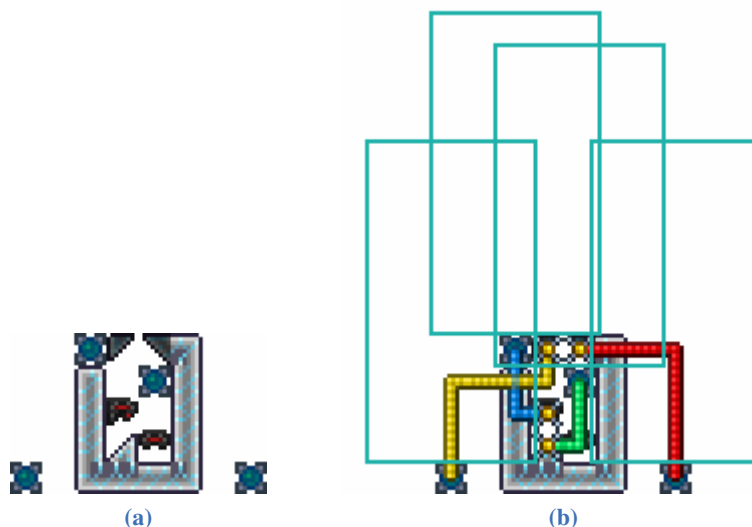


图 6.9: 玩家试图左右移动时会触发玩家感应器并实化半砖将玩家推回；试图上平台时平台虚化，玩家掉落；试图下平台时下方半砖实化将玩家推回。

### 6.2.3 刷怪感应器

刷怪感应器应用于刷怪场和 Boss 战场中，用来检测各种敌怪生成。大多数敌怪都可以直接触发压力板，所以一般情况下压力板就可以用来做刷怪感应器。穿墙怪不会触发压力板，只能使用特殊方法处理。

一个勉强及格的方案是在玩家周围放置压力板，穿墙怪攻击玩家时，玩家被击退，触发压力板，从而敌怪被检测到。这个方案的缺点是明显的，但是暂时也没有更好的方法。

### 6.2.4 树感应器

树感应器用于在树场中检测树的生成。当前景物块上方有树时，该前景物块无法被虚化。因此，可以使用制动器、飞镖机关和青绿压力垫板来检测树的生成。

## 6.3 网线

在设计大型电路时，有时会遇到需要在两地之间传递复杂信号的情况，例如在 A 地设置多个传感器，在 B 地进行处理并响应。一般来说每个信号都需要接一根线，这样一来当信号较复杂时，接线会占用非常大的空间。这是由于一根电线能传递的信号复杂度太低：一根电线只能选择激活或不激活两种状态，所以每根电线只能传递一个二进制位。

在了解了逻辑结算机制以后，我们可以使用一根电线传递多个二进制位。这个功能与现实生活中网线的功能类似，所以我把它称为网线。因为逻辑门是对逻辑帧敏感的，我们可以将多个二进制位在不同的逻辑帧通过一根电线发送出，并在接收端将这些信息解析。归根到底，我们需要做一个编码器和一个解码器。编码器用来把多根线上的简单信



号翻译为一根线上的复杂信号，解码器用来将一根线上复杂信号翻译为多根线上的简单信号。

编码器和解码器的设计取决于信号特点，因此这里不给出固定的做法，仅给出一个简单例子作参考。我们在 A 地放置 8 根火把，使用开关可以改变它们的状态；在 B 地放置 8 根火把；A 地和 B 地之间只能通过 1 根线连接。我们希望在设置完 A 地的火把之后，触发一个开关，就可以把 A 地的火把状态更新到 B 地，即利用一根线传递八个二进制位。

先考虑只有一个火把的情况，首先设置 A 火把，然后触发一个开关，使 B 火把和 A 火把同步。这个装置在数电上叫做 D 触发器。这个电路与置 1/置 0 电路的区别不过它的置 1 或置 0 是通过 A 火把的状态控制。我们只需要稍微修改一下置 1/置 0 电路，就可以用一个逻辑门完成这个功能（图 6.10）。



图 6.10: 左边的开关改变 A 火把的状态，右边的开关将 A 火把状态更新到 B 火把。

那么对于 8 个火把的情况，我们只需要在 8 个逻辑帧中依次触发对应的 D 触发器，就可以把信号发送出去了（图 6.11）。

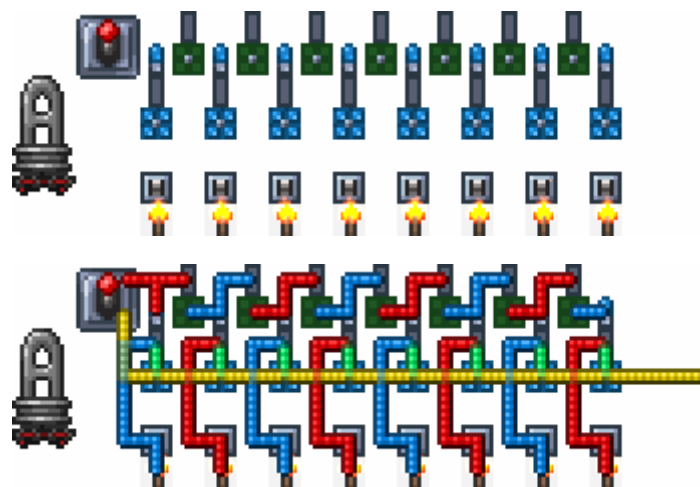
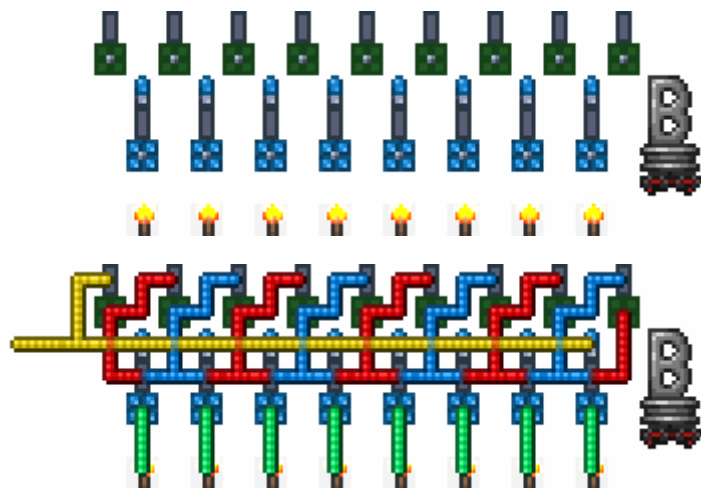


图 6.11: 8 个开关分别控制 8 个火把的状态，控制杆用来发送信号，黄线用来输出。第 0 个逻辑帧黄线激活，用来做启动信号，随后通过上方的逻辑延迟器，8 个 D 锁存器依次在 1~8 个逻辑帧激活。

在 B 地，我们需要把各个逻辑帧的输入提取出来，然后分别输出给 8 个火把（图 6.12）。

以上的装置虽然使用了很多逻辑门，但是当 AB 两地距离较远时，使用这些逻辑门总比使用 8 根电线连接 AB 两地要好。



**图 6.12:** 接收到黄线（在第 0 个逻辑帧）的激活时开始工作。上方的逻辑延迟器依次在 1~8 个逻辑帧将下方对应的有效逻辑灯点亮，对应逻辑帧时如果黄线激活，那么下方对应的火把被激活。第 9 个逻辑帧复位。

## 6.4 传送阵

传送阵，也就是传送机阵列，可以在多个传送机之间传送。根据功能不同，可以分为单向一传多、双向一传多、单向多传多、双向多传多、互传。根据操作方式，可以分为预设、一键和自动。预设指传送前需要通过多个操作设置传送目标；一键指激活一个开关就可以到对应的传送机；自动指不需玩家操作，在需要时自动传送。

阅读本节接下来内容之前请读者先熟悉节 3.10 和小节 2.3.7。

### 6.4.1 利用传送机本身的大小

传送机的大小是 3\*1，因此一个传送机上可以接出两根同色电线，所以简单地就可以做出双向一传八（图 6.13）。

### 6.4.2 利用传送区域大小

一个传送机上的传送区域大小为 3\*3，因此多个传送机的传送区域可能重叠，当玩家站在重叠区域时可以被多个传送机传送，从而增加传送目标数（图 6.14）。

### 6.4.3 用一根电线连接多个传送机

当一根线连接了多个传送机图格时，只会其中的两个图格间传送。至于是在哪两个图格之间传送，则取决于这根电线上的结算顺序，也就是取决于这根电线被激活的位置（具体规则详见[小节 2.3.7](#)）。利用这个特性，可以通过激活一根线上的不同位置来达到在指定传送机之间传送的目的（[图 6.15](#)）。



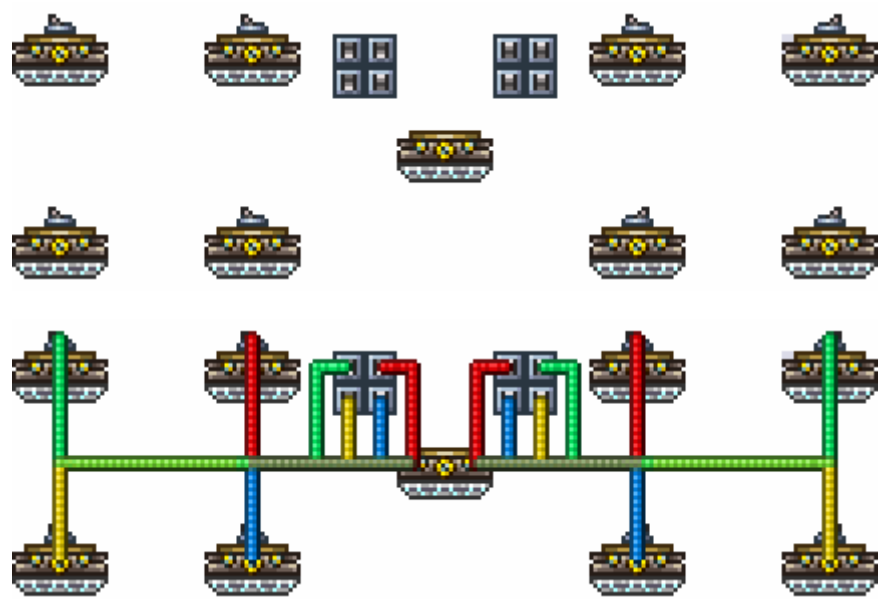


图 6.13: 传送机左边图格可以用四色电线接到四个不同传送机，右边图格可以用四色电线接到另外四个不同传送机。

6.4.4 利用结算顺序

在节 3.10中我们介绍了各种情况下电路的结算顺序。在这里我们将利用这些结算顺序来构造结构更复杂、功能更强大的传送阵。使用电线 a 连接传送机 A 和 B，使用电线 b 连接传送机 B 和 C，如果在电路结算中，a 在 b 之前结算，那么站在传送机 A 上的玩家就会依次传送到 B 和 C。在下一帧中，玩家会在 C 上出现，而 B 上只会出现传送的特效，玩家并不会触发 B 上的任何物理机制，例如受到熔岩伤害、触发加重压力板等。B 在这里起到了中转的作用。利用多级中转可以实现非常复杂的传送功能。

- 首先来看一下如何利用结算顺序做中转（??）。
- 利用传送链可以实现任意传送机互传（??）。
- 利用逻辑结算机制可以大大减少电路占地面积（??）。

6.4.5 注记

传送阵本身的功能是为了在多地之间快速旅行。如果违背了这个原则，那么传送阵的实用价值就要打折扣。例如预设置的传送阵（??）在操作上与简单的手控多级传送??等价，但是建造成本却大大提高。在设计传送阵时不要迷恋表面上的传送数量，而是要结合用户体验和电路面积综合考量。对于更丰富的传送阵思路，请参考<https://www.bilibili.com/video/av24905110/>。

6.5 显示器

显示器从原理上分为分段显示器、密集矩阵显示器、稀疏矩阵显示器、像素盒显示器。



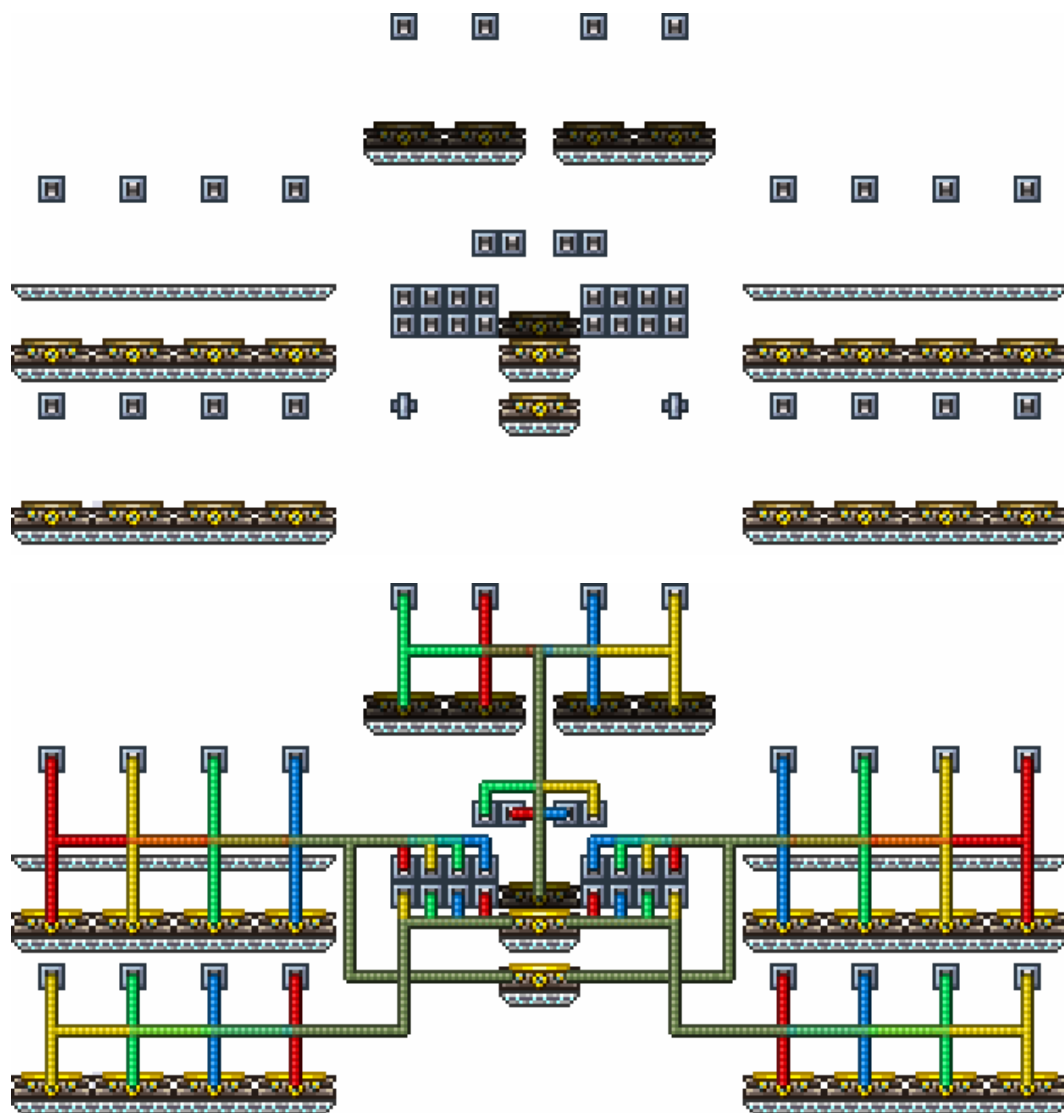


图 6.14: 三个传送机的传送区域有三格重叠，当玩家站在重叠区域时可以传送到共 20 个其他传送机上。

### 6.5.1 分段显示器

分段显示器，指根据显示需要，将显示界面分为若干部分分别控制的显示器。在显示器中有部分光源状态始终同步的情况下，将这些光源当作单个光源处理，即为一段。在十进制数显中我们将显示器分为了七段；在二进制数显中分为了两段；在章 2 的思考题中，将月相显示器的分段任务留给了读者。

分段之后就可以设计控制电路。图 2.24 和图 2.41 中展示了十进制数显的两种不同控制电路风格。前者逻辑门排列紧密，需要更多排换线器；后者逻辑门有间隔，需要换线器更少但是占用空间更大。由于显示部分本身体积就很小，使用占用空间大的控制电路容易导致接线难看（图 2.36）。

另外，图 2.24 和图 2.41 展示了十进制数显的两种不同分段方式。一个分段显示器可以有多种分段方式，不同的分段方式对应不同的控制电路和接线。一个自然的问题是，是否可以设计出一个分段方式使得控制电路最简？答案是，没有一个固定的套路可以使

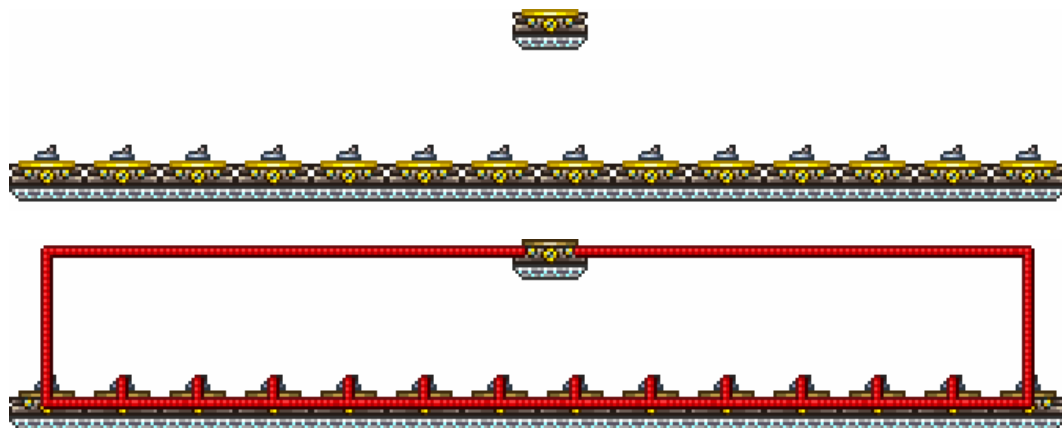


图 6.15: 单向多传一。无论激活下面哪个传送机上的开关, 第一个结算的传送机图格都是该开关正下方的图格, 最后一个结算的传送机都是上方传送机的左图格或右图格。因此激活开关时会在上方传送机和下方对应传送机之间传送。

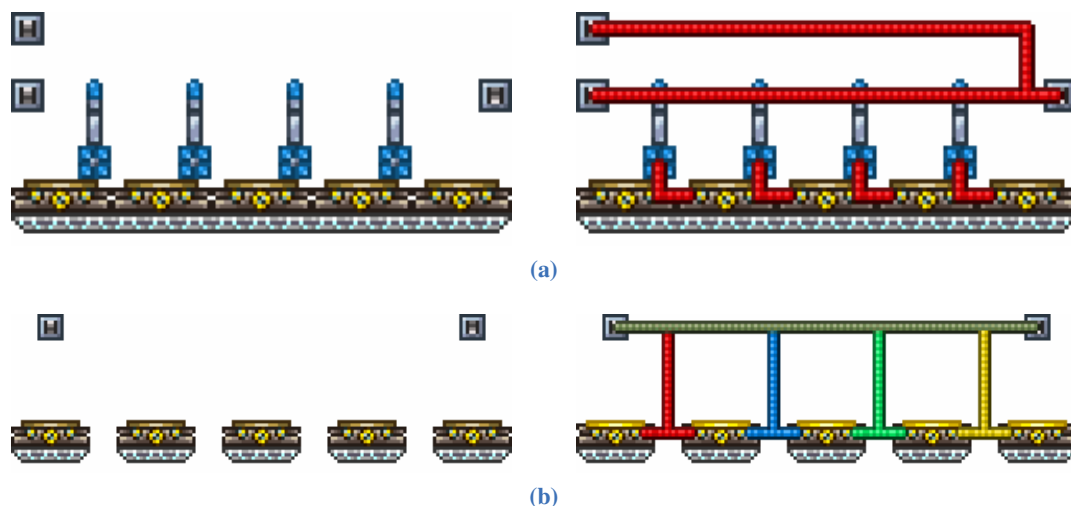


图 6.16: ((a))人物站在左边传送机上, 右击左下开关, 四个逻辑门从左到右依次激活, 人物被依次传送到最右边; 人物站在右边传送机上, 右击右下开关或左上开关, 四个逻辑门从右到左依次激活, 人物被依次传送到最左边; ((b))原理与((a))类似, 只不过是利用了红蓝绿黄依次结算。

控制电路最简。尽管如此, 我们往往还是可以做出部分简化。下面以一个例子来介绍优化分段方式的方法。

我们希望修改十进制数显的分段来减少控制电路的大小。数显初始分段如图 2.22 所示。在这个分段下, 数字与分段的对应关系如表 6.1。

上面的矩阵在  $\mathbb{F}_2^3$  下的秩为  $7^4$ , 所以显示器至少要分成 7 段, 没有办法减少段数。那么就没有可能减少控制电路中的换线器数量了吗? 有的。在某些情况下, 可以将同色的两段线连接到同一排换线器上并且使其不重叠 (??)。用这种方法, 7 段线至多可以合并三对, 这样就可以接在一行中。

我们对对应矩阵进行初等列变换, 尝试将两列中的 1 分离。每列的列标是集合  $\{a, b,$

<sup>3</sup>域  $\mathbb{F}_2$  包含 0, 1 两个元素, 可以将 0 看作偶数, 1 看作奇数。加法规则:  $0+0=1+1=0$ ,  $0+1=1+0=1$ ; 乘法规则:  $0*0=0*1=1*0=0$ ,  $1*1=1$ 。

<sup>4</sup>后文中会介绍该矩阵的初等列变换操作并证明矩阵的列秩为 7。

	a	b	c	d	e	f	g
0	1	1	1	0	1	1	1
1	0	0	1	0	0	1	0
2	1	0	1	1	1	0	1
3	1	0	1	1	0	1	1
4	0	1	1	1	0	1	0
5	1	1	0	1	0	1	1
6	1	1	0	1	1	1	1
7	1	0	1	0	0	1	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1

表 6.1: 十进制数显传统接线的显示矩阵

c, d, e, f, g} 的子集，列标为 X 的列的内容称为列 X。初等列变换分为两种操作：

1. 交换列 X 和列 Y，同时交换 X 和 Y；
2. 将列 X 加到列 Y 上 ( $\mathbb{F}_2$  加法)，同时将 X 改为 X 和 Y 的对称差<sup>5</sup>。

初等列变换过程如图 6.17 和图 6.18。图 6.17 使靠右的列的上部为 0，经过这一步可以看出来矩阵的列秩为 7；图 6.18 使靠左的列的下部为 0。经过变换后，第 1 列可以和第 5 列合并，第 2 列可以和第 6 列合并，第 3 列可以和第 7 列合并。合并后接线如??，注意由于合并的两列有重复段，在显示屏上不能使用同色电线，必须要使用额外的换线器。继续使用初等列变换可以减少换线器使用 (??)。

在上面的例子中，显示器在二进制输入为 1010~1111 时会熄灭，也就是说显示器有十一种显示状态。如果我们不需要熄灭状态，减少一种状态，看看是否可以进一步减少换线器使用。

这里使用另一种技巧，我们先做一种预处理来减少矩阵中 1 的数量，这样有利于分离。定义矩阵列的反向操作：将该列列标的大小写互换，将该列 01 互换。标有大写的分段在接线时，其火把的 01 状态与正常情况相反。显然这个操作只有在初等列变换之前进行才有意义。

在初始的显示矩阵中，除了 e 列的 1 比 0 少，其他所有列的 1 都比 0 多，所以将其其他所有列反向，可以减少矩阵中 1 的个数 (表 6.2)。对这个矩阵进行化简得到??，再作一些技巧上的处理 (??)，就可以不使用换线器 (??)。

此外，还有一些方法可以作为化简手段：

- 交换矩阵的行。在之前的操作中，控制 0~9 的十个逻辑门是按照数字顺序排列的。在减少可读性的情况下，可以改变它们的顺序，而改变它们的顺序可能会使矩阵的列分离开。因为顺序改变了，所以由递次电路控制的数显不能用这种方法。
- 将某一段用两根线控制。7 段线合并 3 对可以放在一行内，8 段线合并 4 对也可以放在一行内。这样的话可以将显示矩阵中有较多 1 的一段分成有较少 1 的两段，可能有利于分离。
- 将多于两段合并。这种方法的局限性较强，因为它影响多层控制电路的接线。如果最终可以只化为 1 层，那么这种方法可用。

<sup>5</sup>集合 A 与集合 B 的对称差定义为  $A \Delta B = (A \cup B) - (A \cap B)$

	a	b	c	d	e	f	g
0	1	1	1	0	1	1	1
1	0	0	1	0	0	1	0
2	1	0	1	1	1	0	1
3	1	0	1	1	0	1	1
4	0	1	1	1	0	1	0
5	1	1	0	1	0	1	1
6	1	1	0	1	1	1	1
7	1	0	1	0	0	1	0
8	1	1	1	1	1	1	1
9	1	1	1	1	0	1	1

(a) 原矩阵

	abcefg	c	b	d	e	f	g
0	1	0	0	0	0	0	0
1	0	1	0	0	0	1	0
2	1	0	1	1	0	1	0
3	1	0	1	1	1	0	0
4	0	1	1	1	0	1	0
5	1	1	0	1	1	0	0
6	1	1	0	1	0	0	0
7	1	0	1	0	1	0	1
8	1	0	0	1	0	0	0
9	1	0	0	1	1	0	0

(b) 将第 1 列加到第 2,3,5,6,7 列, 交换第 2,3 列

	abcefg	cf	b	d	e	f	g
0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0
2	1	0	1	1	0	1	0
3	1	0	1	1	1	0	0
4	0	1	1	1	0	0	0
5	1	1	0	1	1	1	0
6	1	1	0	1	0	1	0
7	1	0	1	0	1	0	1
8	1	0	0	1	0	0	0
9	1	0	0	1	1	0	0

(c) 将第 2 列加到第 6 列

	abcefg	cf	bdf	e	d	f	g
0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0
2	1	0	1	0	0	0	0
3	1	0	1	1	0	1	0
4	0	1	1	0	0	1	0
5	1	1	0	1	1	1	0
6	1	1	0	0	1	1	0
7	1	0	1	1	1	1	1
8	1	0	0	0	1	0	0
9	1	0	0	1	1	0	0

(d) 将第 3 列加到第 4,6 列, 交换第 4,5 列

	abcefg	cf	bdf	ef	f	d	g
0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0
2	1	0	1	0	0	0	0
3	1	0	1	1	0	0	0
4	0	1	1	0	1	0	0
5	1	1	0	1	0	1	0
6	1	1	0	0	1	1	0
7	1	0	1	1	0	1	1
8	1	0	0	0	0	1	0
9	1	0	0	1	1	1	0

(e) 将第 4 列加到第 6 列, 交换第 5,6 列

图 6.17



	abcefg	cf	bdf	ef	f	abcdfg	g		abcefg	cf	bdf	ef	e	abcdfg	g
0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0
2	1	0	1	0	0	0	0	2	1	0	1	0	0	0	0
3	1	0	1	1	0	0	0	3	1	0	1	1	0	0	0
4	0	1	1	0	1	0	0	4	0	1	1	1	1	0	0
5	0	1	0	0	1	1	0	5	0	1	0	1	1	1	0
6	0	1	0	1	0	1	0	6	0	1	0	1	0	1	0
7	0	0	1	0	1	1	1	7	0	0	1	1	1	1	1
8	0	0	0	1	1	1	0	8	0	0	0	0	1	1	0
9	0	0	0	0	0	1	0	9	0	0	0	0	0	1	0

(a) 将第 6 列加到第 1,4,5 列

(b) 将第 5 列加到第 4 列

	abcefg	cf	bdf	ef	e	abcdfg	acf		abcefg	cf	bdf	ce	e	abcdfg	acf
0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0
2	1	0	1	0	0	0	0	2	1	0	1	0	0	0	0
3	1	0	1	1	0	0	0	3	1	1	1	1	0	0	0
4	0	1	1	1	1	0	0	4	0	0	1	1	1	0	0
5	0	1	0	1	1	1	0	5	0	0	0	1	1	1	0
6	0	1	0	1	0	1	0	6	0	0	0	1	0	1	0
7	0	0	0	0	0	0	1	7	0	0	0	0	0	0	1
8	0	0	0	0	1	1	0	8	0	0	0	0	1	1	0
9	0	0	0	0	0	1	0	9	0	0	0	0	0	1	0

(c) 将第 7 列加到第 3,4,5,6 列

(d) 将第 4 列加到第 2 列

	abcefg	abeg	bdf	ce	e	abcdfg	acf
0	1	0	0	0	0	0	0
1	1	1	0	0	0	0	0
2	1	0	1	0	0	0	0
3	0	1	1	1	0	0	0
4	0	0	1	1	1	0	0
5	0	0	0	1	1	1	0
6	0	0	0	1	0	1	0
7	0	0	0	0	0	0	1
8	0	0	0	0	1	1	0
9	0	0	0	0	0	1	0

(e) 将第 2 列加到第 1 列

图 6.18

	a	b	c	d	e	f	g
0	0	0	0	1	1	0	0
1	1	1	0	1	0	0	1
2	0	1	0	0	1	1	0
3	0	1	0	0	0	0	0
4	1	0	0	0	0	0	1
5	0	0	1	0	0	0	0
6	0	0	1	0	1	0	0
7	0	1	0	1	0	0	1
8	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0

表 6.2

### 6.5.2 密集矩阵显示器

用单色光源可以实现至少有一边长度至多为 4 的密集显示器（图 6.19）。

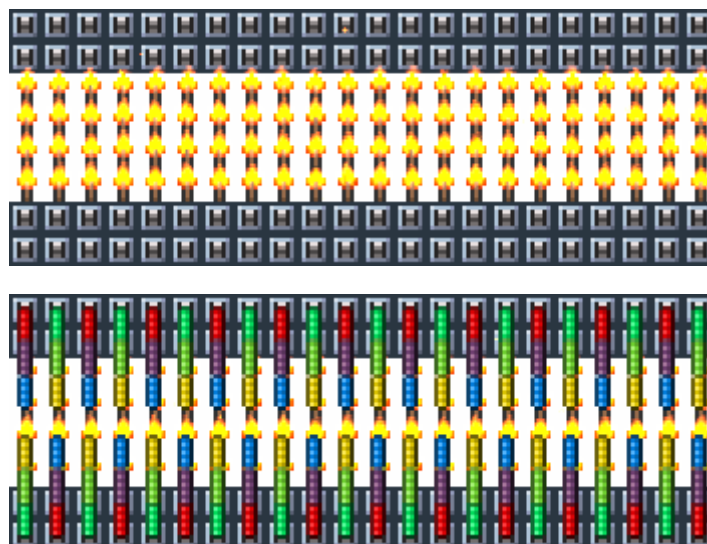


图 6.19: 第一行开关控制第一行火把，第二行开关控制第二行火把，第三行开关控制第三行火把，第四行开关控制第四行火把。

### 6.5.3 稀疏矩阵显示器

稀疏矩阵显示器主要有两个参数：每个像素的光源面积与占用面积。稀疏矩阵显示器的大小不受限，一个典型的稀疏矩阵显示器如图 6.20 所示，当红线和蓝线在同一个逻辑帧输入时，火把响应，切换状态。显示器更新时，各行的红线逐次在各个逻辑帧激活，每列的蓝线在该逻辑帧选择性激活，用来控制该行的显示。这个例子中每个光源面积是 4，占用面积是 12，简称 4 占 12，或 4/12。目前使用这种显示器最杰出的作品是<https://www.bilibili.com/video/av22343683>。

在这里我们看到控制每个像素的模块是一个占用 3 格的故障逻辑门。目前认为这是最小的控制单元<sup>6</sup>。发光面积之间的间隔用来让控制的电线穿过。如果可以把火把摆在控

<sup>6</sup>期待有人能开发出更小的显示器

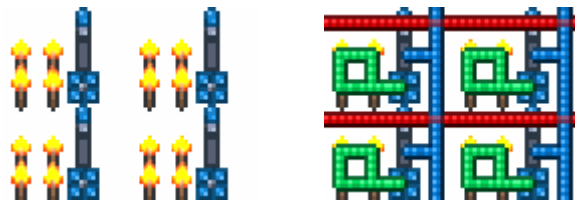


图 6.20: 典型的矩阵显示器

制的电线上，那么就可以让显示器更紧凑（图 6.21）。

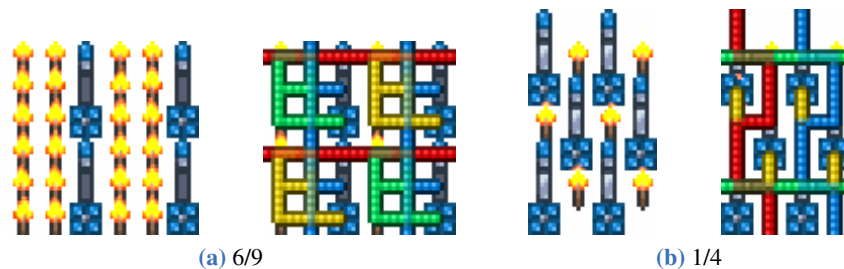


图 6.21: 紧凑的矩阵显示器

到这里你也许会有疑问，如果火把被摆在控制的电线上，不就会被控制的电线干扰吗？是的，是会被干扰，但是我们可以通过额外的激活来抵消这个干扰。具体说来，我们可以记录下每根控制电线激活的次数。如果一根控制电线激活了偶数次，那么它经过的火把是没有被干扰的。如果激活了奇数次，那么它经过的火把被取反，这时我们额外再激活一次这根线，就可以将火把恢复到没被干扰的状态。

接下来需要处理的一个问题是，额外激活的一次会不会干扰到某个像素？显然连接有效逻辑灯的电线不会，连接故障逻辑灯的电线可能会。事实上，如果我们先激活连接有效逻辑灯的电线，那么由于每根线都激活了偶数次，所有有效逻辑灯都会被关闭，此时再激活连接故障逻辑灯的电线不会干扰到任何一个像素。

此外，如果你足够细心，你会发现图 6.21(b)中连接故障逻辑灯的电线也连接了有效逻辑灯，这不会出 bug 吗？在这种情况下，只要将有效逻辑灯的默认状态设为亮就行了。

还有一个非常巧妙的办法可以减少显示单元的面积，那就是使用小地图。半透明的小地图中每格 2.5 像素，在屏幕上 2 像素和 3 像素交替排列。有很多使用小地图显示的例子：

- 【Terraria】做贪吃蛇—TheRedstoneCrafter <https://www.bilibili.com/video/av32265379>
- 在 Terraria 中玩俄罗斯方块!? <https://www.bilibili.com/video/av38924330>

#### 6.5.4 像素盒显示器

使用像素盒可以实现宽至多为 24 的密集矩阵显示器。因为其电路复杂，这里不给出电路图，仅给出原理。

首先来看如何更新屏幕状态。根据像素盒特性，显然每个像素盒都需要两个方向各一条线来控制，其中横向的线激活才会导致像素盒响应，仅纵向激活是不会响应的，这样一来屏幕的每行是独立的，可以逐行更新。

然后来看如何更新一行。当一行中的横向电线激活时，需要点亮的像素盒上的纵向电线必须被同一个电源激活，这意味着每个横向电线只能控制至多 3 个像素盒（图 6.22）。



图 6.22: 八个开关分别将像素盒状态变为 000, 001, 010, 011, 100, 101, 110, 111。

幸亏像素盒本身就有分线盒的作用，这样一来每行可以安排八个横向电线，更新这一行时从中间向两侧，每三个为一组更新（图 6.23）。



图 6.23: 更新一行时，左边 12 个像素盒以 3 个为一组从右向左更新，右边 12 个像素盒以 3 个为一组从左向右更新。

## 6.6 自动化

## 6.7 思考题

### 第 6 章 思考题

1. 为什么在分段显示器的化简中对矩阵进行初等列变换时需要对列标做对称差？
2. 为什么两边长都大于 4 的非像素盒密集矩形随机显示器不存在？
3. 为什么像素盒显示器的宽度至多为 24？

## 第 7 章 农场

前面我们主要围绕电路理论来讲。这一章中我们会看到一些利用游戏机制的精妙设计。

### 7.1 水晶碎块农场

#### 7.1.1 水晶碎块的生成机制

在**困难模式**中，游戏每一帧会将如下的步骤执行多次，执行次数 = 世界宽度 × 世界高度 × 0.000015：

1. **选取支撑块**。在一个超大的矩形区域内随机取一格，如果恰好是粉冰雪块或珍珠石块，并且取的这一格在洞穴或地狱，那么有 1/110 的概率可以进入下一步，否则不生成水晶。这个矩形区域左右端距离世界左右端各 10 格，上端为地表，下端距离世界下端 20 格。
2. **选取生成格**。把上一步中取到的一格叫做支撑块。在支撑块下左右三个相邻格中随机取一格，其中取下的概率是 1/2，左右的概率各 1/4，这一格就作为水晶的生成格。生成格不能被其他图格阻挡，否则不能生成水晶。
3. **局部数量限制**。统计以支撑块为中心，13 格 × 13 格的正方形区域内已经存在的水晶碎块数量，如果超过了 1，那么不能生成水晶。
4. **生成水晶**。如果前三步所有检查均通过，那么在生成格生成水晶。

#### 7.1.2 三大设计要点

一个水晶农场主要分三个部分：生长、收割、收集。在生长环节，需要选取使用粉冰雪块还是珍珠石块，还需要设计方块的摆放方式。收割可以通过简单的虚化实现，但是我们需要知道收割的频率。如果使用相对高频的驱动进行收割，那么比较消耗 CPU 资源；如果使用过于低频的驱动进行收割，那么容易达到局部上限。在收集环节，需要选取使用半砖、传送带还是传送机。半砖速度快，稳定性差；传送带速度慢，稳定性好；传送机速度快，稳定性快，但是不方便操作。所有这三个部分的方案设计都是通过计算实现的。

要生成水晶碎块，需要通过四层检查：选取的方块恰好可以做支撑块；1/110 的概率；生成格不能被阻挡；局部数量限制。其中 1/110 的概率是固定值，先不管。局部数量限制是与基础生成速度相关的，如果水晶碎块生成得快，那么就很容易达到局部数量上限，否则可以忽略掉局部数量上限。

选取方块恰好可以做支撑块的概率，是矩形区域内，洞穴和地狱中粉冰雪块和珍珠石块的比例。我们先假设这个比例是 100%。生成格不能被阻挡，意味着支撑块不能过于密集，不过我们先不管，先假设这个概率也是 100%。这样的话，水晶碎块的最大生

成速率是 (世界宽度  $\times$  世界高度  $\times 0.000015$ ) 个/110 帧，这是整个世界中水晶碎块的速率。那么每格上水晶碎块的速率就是 0.000015 个/110 帧<sup>1</sup>。13 $\times$ 13 区域的生成速率就是  $0.000015 \times 13^2 = 0.002535$  个/110 帧。达到局部生成上限平均需要 2 个/(0.002535 个/110 帧)=86785 帧，大约是 1 天。

回到前两个概率。前两个概率合起来，就是任选一个方块的一个表面，可以生成水晶的概率。换句话说，生成速率与可生成的表面积成正比。??所示的排列方式可以达到最大的表面积，在这个排列下，任选一个方块，可以做支撑块的概率是 1/2，生成格不被阻挡的概率是 100%，所以水晶碎块的生成速率全部要除以 2，达到局部生成上限平均需要约 2 天。收割周期确定为 1 天左右，这样不会过于频繁，也不会有太多方块达到上限。

然后考虑收割方式，是一次性虚化收割，还是一块一块收割？掉落物上限是 400，一天收割一次的话，要求水晶农场的面积控制在  $13 \times 13 \times 400$  格 = 67600 格以内。如果规模更大，就需要一块一块收割。这里每块的大小设置取决于收割后运输的速度。假设总共有  $N$  块，那么每块的收割间隔是  $(86400/N)$  帧。假设总面积是  $S$  格，那么每块的面积是  $(S/N)$  格，每块收割时掉落约  $(S/N/169)$  个水晶碎块。达到 400 个水晶碎块，需要收割  $400/(S/N/169) = (67600N/S)$  块，也就是说，水晶从掉落到收集，时间是  $(86400/N) \times (67600N/S) = (5840640000/S)$  帧，超过这个时间就很有可能因为掉落物上限损失水晶。对于大世界来说， $S$  至多是  $8400 \times 2400 = 20160000$  格，时间大约是 290 帧，不到 5 秒。在 5 秒内收集全世界的水晶是不可能的。

水晶农场面积越小，对收集速度的要求就越低。当规模较小时，可以使用??最大化生成速率；当规模较大时，需要优先考虑便于半砖与传送器运输的结构以增加收集速度。

---

<sup>1</sup>与 worldSurface 和 rockLayer 占世界高度比例相关，这里是非常粗略的估算

## 7.2 全自动树场

## 7.3 生命果/世花农场

## 7.4 液体反应池

## 7.5 松露虫农场

## 7.6 采沙场

## 7.7 天梯神教

### 7.7.1 南瓜月天梯神教

### 7.7.2 霜月天梯神教

## 7.8 南瓜神教

## 7.9 全自动月亮事件

## 7.10 DPS 纪录

## 7.11 速度纪录

## 7.12 Boss 速杀场地



## 附录 环境判定

多数环境判定中包含对屏幕附近方块的计数。这个范围与游戏窗口分辨率有关，但是目前关系尚未明确。

环境名称	判定规则
四柱区域	玩家中心到四柱中心距离不大于 4000 像素
地牢	玩家中心在 worldSurface 之下，玩家中心在地牢墙前，屏幕附近地牢砖格数至少为 250
地下沙漠	玩家中心坐标大于 3200，玩家中心在各种沙岩墙或各种硬化沙墙前，屏幕附近沙漠方块格数大于 1000
沙尘暴	玩家中心所在格坐标不在 worldSurface 之下，在沙漠环境，玩家不在海洋环境，正在发生沙尘暴
丛林	屏幕附近丛林方块格数至少为 80
腐化之地	屏幕附近腐化方块格数-5× 屏幕附近向日葵格数-屏幕附近神圣方块格数至少为 200
猩红之地	屏幕附近猩红方块格数-5× 屏幕附近向日葵格数-屏幕附近神圣方块格数至少为 200
神圣之地	屏幕附近神圣方块格数 +10× 屏幕附近向日葵格数-屏幕附近腐化方块格数-屏幕附近猩红方块格数至少为 100
雪原	屏幕附近雪原方块格数至少为 300
陨石	屏幕附近陨石方块格数至少为 50
沙漠	屏幕附近沙漠方块格数大于 1000
蘑菇	屏幕附近蘑菇方块格数大于 100
旧日军团	玩家中心到埃特尼亚水晶中心距离不大于 4000 像素

## 附录 刷怪机制

执行刷怪的函数为 NPC.SpawnNPC()。

如果 NPC.noSpawnCycle 为 true，那么这一帧不刷怪，把 NPC.noSpawnCycle 重置为 false。

如果要刷怪的话，会对每个未死亡的玩家执行刷怪过程。在史莱姆雨进行时，会在其他刷怪之前插入史莱姆雨的刷怪。史莱姆雨的刷怪是额外刷怪，不会影响正常刷怪。如果有一个玩家成功进行了正常刷怪，那么跳过剩余玩家的刷怪。

**注** 每帧中，除史莱姆雨刷怪外，至多只会在一个玩家周围进行一次成功刷怪。

进行正常刷怪时，首先要判断刷怪率和刷怪量，然后决定是否要刷怪，然后决定刷怪点和刷怪面，最后决定刷什么怪。

### Contents

<b>B.1</b>	<b>刷怪率和刷怪量</b>	<b>101</b>
<b>B.2</b>	<b>刷怪点和刷怪面</b>	<b>103</b>
<b>B.3</b>	<b>刷怪类型</b>	<b>103</b>
B.3.1	事件刷怪	103
B.3.2	禁止穿墙刷怪	103
B.3.3	小动物刷怪	104
B.3.4	太空刷怪	104
B.3.5	水中刷怪	104
B.3.6	大理石刷怪与花岗岩刷怪	104
B.3.7	蜘蛛巢刷怪和地下沙漠刷怪	104
<b>B.4</b>	<b>刷怪失败</b>	<b>105</b>
<b>B.5</b>	<b>刷怪种类</b>	<b>106</b>
B.5.1	四柱	106
B.5.2	太空刷怪	107
B.5.3	事件刷怪	108
B.5.4	昏迷男子	110
B.5.5	蜘蛛巢	110
B.5.6	地下沙漠	111
B.5.7	巨骨舌鱼	112
B.5.8	血水母	112
B.5.9	嗜血怪	112
B.5.10	海洋	112
B.5.11	沉睡渔夫	113

B.5.12 食人鱼	113
B.5.13 蓝水母	113
B.5.14 水中小动物	113
B.5.15 受缚哥布林	113
B.5.16 受缚巫师	114
B.5.17 小动物	114
B.5.18 地牢	115
B.5.19 陨石	116
B.5.20 撒旦军队	116
B.5.21 霜月	116
B.5.22 南瓜月	118
B.5.23 日食	120
B.5.24 发光蘑菇地	122
B.5.25 腐地蠕虫	122
B.5.26 宝箱怪	122
B.5.27 幻灵 (ID:82)	123
B.5.28 弹跳杰克南瓜灯 (ID:304)	123
B.5.29 骷髅博士 (ID:52)	123
B.5.30 紫胶虫 (ID:219)	123
B.5.31 地下蠕虫	123
B.5.32 老鼠	123
B.5.33 蜗牛 (ID:359)	123
B.5.34 丛林青蛙	123
B.5.35 困难模式丛林	124
B.5.36 丛林	124
B.5.37 沙尘暴	125
B.5.38 木乃伊	125
B.5.39 地表神圣	125
B.5.40 附魔剑 (ID:84)	125
B.5.41 猩红之地	125
B.5.42 腐化之地	126
B.5.43 地表综合	126
B.5.44 地表白天	126
B.5.45 地表夜晚	127
B.5.46 地下	128
B.5.47 地狱	129
B.5.48 洞穴	129

<b>B.6 史莱姆雨刷怪</b>	<b>130</b>
<b>B.7 特殊判定</b>	<b>131</b>
B.7.1 沙块的局部聚簇检查	131
B.7.2 Boss 与事件检查	131

## B.1 刷怪率和刷怪量

刷怪率用于控制刷怪速度，刷怪量用于控制刷怪上限。

如果玩家的活跃敌怪数量达到了刷怪量，那么刷怪失败。活跃敌怪数量是以玩家为中心，宽为  $2activeRangeX$ ，高为  $2activeRangeY$  的矩形中的 NPC 的加权和。不同 NPC 权重不同<sup>1</sup>。

刷怪率为  $n$ ，意味着每次判定时的基础刷怪概率为  $1/n$ 。

刷怪率基础值为 600，刷怪量基础值为 5。接下来会按照下面步骤修改刷怪率和刷怪量。注意，列表中的各项不互斥：如果同时满足两项的条件，那么两项都要结算，结算顺序与描述顺序相同。

1. 在困难模式中，刷怪率改为 540，刷怪量改为 6。
2. 如果玩家纵坐标距离世界底端小于 200 格，刷怪量乘 2。如果玩家纵坐标在  $rockLayer+sHeight$  之下且距离世界底端不小于 200 格，刷怪率乘 0.4 向下取整，刷怪量乘 1.9 向下取整。如果玩家纵坐标不在  $rockLayer+sHeight$  之下且在  $worldSurface+sHeight$  之下，刷怪率乘 0.5（困难模式为 0.45）向下取整，刷怪量乘 1.7（困难模式为 1.8）向下取整。如果玩家纵坐标不在  $worldSurface+sHeight$  之下：
  - 如果在晚上，刷怪率乘 0.6 向下取整，刷怪量乘 1.3 向下取整。
  - 如果在血月，刷怪率乘 0.3 向下取整，刷怪量乘 1.8 向下取整。该效果会与晚上叠加。
  - 如果在南瓜月或霜月，并且玩家纵坐标在  $worldSurface$  之上，那么刷怪率乘 0.2 向下取整，刷怪量乘 2。
  - 如果在日食，刷怪率乘 0.2 向下取整，刷怪量乘 1.9 向下取整。
3. 如果在苔原环境，并且玩家纵坐标在  $worldSurface$  之上，刷怪量乘  $(1+Main.cloudAlpha)$  并向下取整，刷怪率乘  $(1 - Main.cloudAlpha/2)$  并向下取整。
4. 如果在地牢环境，刷怪率乘 0.4 向下取整，刷怪量乘 1.7 向下取整。如果不在地牢环境并且在沙尘暴环境，刷怪率乘 0.9（困难模式为 0.4）向下取整，刷怪量乘 1.2（困难模式为 1.5）向下取整。如果不在沙尘暴环境并且在地下沙漠环境，刷怪率乘 0.3（困难模式为 0.2）向下取整，刷怪量乘 2。如果不在以上三种环境并且在丛林环境，刷怪率乘 0.4 向下取整，刷怪量乘 1.5 向下取整。如果不在以上四种环境并且在腐化或血腥环境，刷怪率乘 0.65 向下取整，刷怪量乘 1.3 向下取整。如果不在以上六种环境并且在陨石环境，刷怪率乘 0.4 向下取整，刷怪量乘 1.1 向下取整。

<sup>1</sup>ToDo: 权重表

5. 如果在神圣环境，并且玩家纵坐标在 `rockLayer+sHeight` 之下，刷怪率乘 0.65 向下取整，刷怪量乘 1.3 向下取整。
6. 如果世界中有血肉墙并且玩家纵坐标距离世界底端小于 200 格，刷怪率乘 3，刷怪量乘 0.3 向下取整。
7. 如果玩家的活跃敌怪数量小于刷怪量的  $\frac{1}{5}$ ，刷怪率乘 0.6 向下取整。如果玩家的活跃敌怪数量不小于刷怪量的  $\frac{1}{5}$  但是小于刷怪量的  $\frac{2}{5}$ ，刷怪率乘 0.7 向下取整。如果玩家的活跃敌怪数量不小于刷怪量的  $\frac{2}{5}$  但是小于刷怪量的  $\frac{3}{5}$ ，刷怪率乘 0.8 向下取整。如果玩家的活跃敌怪数量不小于刷怪量的  $\frac{3}{5}$  但是小于刷怪量的  $\frac{4}{5}$ ，刷怪率乘 0.9 向下取整。
8. 如果玩家纵坐标在 `worldSurface` 与 `rockLayer` 的中点之下，或者玩家在腐化或血腥环境：
  - 如果玩家的活跃敌怪数量小于刷怪量的  $\frac{1}{5}$ ，刷怪率乘 0.7 向下取整。
  - 如果玩家的活跃敌怪数量不小于刷怪量的  $\frac{1}{5}$  但是小于刷怪量的  $\frac{2}{5}$ ，刷怪率乘 0.9 向下取整。
9. 如果玩家有镇静药剂 buff，刷怪率乘 1.3 向下取整，刷怪量乘 0.7 向下取整。如果玩家有向日葵 buff，刷怪率乘 1.2 向下取整，刷怪量乘 0.8 向下取整。如果玩家有战斗药水 buff，刷怪率乘 0.5 向下取整，刷怪量乘 2。如果玩家有水蜡烛 buff 并且没有和平蜡烛 buff，刷怪率乘 0.75 向下取整，刷怪量乘 1.5 向下取整。如果玩家有和平蜡烛 buff，刷怪率乘 1.3 向下取整，刷怪量乘 0.7 向下取整。如果玩家在水蜡烛区域内，并且在太空，刷怪率乘 0.5 向下取整。
10. 如果刷怪率低于 60，重置为 60。如果刷怪量高于 15，重置为 15。在南瓜月/霜月中（要求玩家纵坐标在 `worldSurface` 之上），刷怪率为 20，刷怪量为  $5 \times (2 + \text{玩家数量})$ 。在撒旦军团事件中，刷怪率为 600，刷怪量为 5。判定为事件刷怪，刷怪率为 20，刷怪量为  $5 \times (2 + \text{玩家数量})$ 。在骷髅王前的地牢环境，刷怪率为 10。
11. 如果判定为小动物刷怪：
  - 如果玩家中心距离世界底端小于 200 格，刷怪量乘 0.5 向下取整。
  - 如果玩家中心距离世界底端不小于 200 格，刷怪量乘 0.6 向下取整。
12. 未判定为事件刷怪，并且不在血月、南瓜月、霜月、日食、地牢环境、腐化环境、血腥环境、陨石环境、撒旦军团事件中，并且未判定为小动物刷怪，考虑玩家的活跃城镇 NPC 数量<sup>2</sup>与玩家中心到世界底端的距离：

	距离小于 200 格	距离不小于 200 格
1 个活跃城镇 NPC	刷怪率乘 1.25 向下取整	刷怪率乘 2 向下取整
2 个活跃城镇 NPC	刷怪率乘 1.5 向下取整	刷怪率乘 3 向下取整
≥3 个活跃城镇 NPC	刷怪率乘 2	刷怪量乘 0.6 向下取整

<sup>2</sup>活跃城镇 NPC 数量是以玩家为中心，宽为 `2sWidth`，高为 `2sHeight` 的矩形内城镇 NPC 的加权和

## B.2 刷怪点和刷怪面

在通过了  $\frac{1}{\text{刷怪率}}$  的基础刷怪概率后，会选取刷怪位置。刷怪位置的选取有一定随机性，如果选取失败就重新选取，直到选取成功为止。如果选了 50 次还没有成功，这次刷怪就取消。选取刷怪位置的过程中，首先要选取一个刷怪点，然后根据刷怪点计算出刷怪面，刷怪位置就在刷怪面上面 1 格。

刷怪点是在一个矩形范围内随机选取的，这个矩形中图格的横坐标从玩家横坐标-生成区域宽度取到玩家横坐标+生成区域宽度，纵坐标从玩家纵坐标-生成区域高度取到玩家纵坐标+生成区域高度。如果矩形范围超出了世界边界，那么按照世界边界对矩形进行裁剪。

如果刷怪点处有未虚化的实体块，或者刷怪点处有人工背景墙，本次选取失败。

进行**太空刷怪判定**。如果通过了太空刷怪判定，刷怪面就是刷怪点。如果未通过太空刷怪判定，从刷怪点向下搜索到第一个未虚化的实体块作为刷怪面；如果刷怪面在安全区域内，本次选取失败。

如果刷怪面上方 3 格 × 3 格区域有部分超出了世界边界，或者其内部有未虚化的实体块或熔岩，本次选取失败。

## B.3 刷怪类型

### B.3.1 事件刷怪

- 玩家纵坐标在 `worldSurface+sHeight` 之上，玩家距离事件中心的横坐标距离小于 3000 像素，那么判定为事件刷怪。
- 玩家纵坐标在 `worldSurface+sHeight` 之上，事件中心到世界中心的横坐标距离不超过 5 格，玩家到  $n$  个城镇 NPC 的最近横坐标距离小于 3000 像素，那么有  $1 - (2/3)^n$  的概率判定为事件刷怪。
- 玩家在四柱区域内，判定为事件刷怪。

### B.3.2 禁止穿墙刷怪

- 未判定为事件刷怪，玩家不在血月、南瓜月、霜月、日食、地牢环境、腐化环境、血腥环境、陨石环境、撒旦军团事件中，玩家中心距离世界底端不小于 200 格，判定禁止穿墙刷怪。
- 未判定为事件刷怪，玩家不在血月、南瓜月、霜月、日食、地牢环境、腐化环境、血腥环境、陨石环境、撒旦军团事件中，玩家中心距离世界底端小于 200 格，对应于玩家的活跃城镇 NPC 数量，分别有 1/2 概率（1 个活跃城镇 NPC）、3/4 概率（2 个活跃城镇 NPC）、9/10 概率（≥3 个活跃城镇 NPC）判定禁止穿墙刷怪。
- 玩家中心在人工背景墙前，判定禁止穿墙刷怪。

B.3.3 小动物刷怪

未判定为事件刷怪，并且不在血月、南瓜月、霜月、日食、地牢环境、腐化环境、血腥环境、陨石环境、撒旦军团事件中，玩家的活跃城镇 NPC 数量与玩家中心到世界底端的距离会决定小动物刷怪的概率：

	距离小于 200 格	距离不小于 200 格
1 个活跃城镇 NPC	1/10	1/3
2 个活跃城镇 NPC	1/5	1/2
≥3 个活跃城镇 NPC	1/3	专家模式 29/30，普通模式一定

B.3.4 太空刷怪

太空刷怪是在刷怪面选取过程中判定的。太空刷怪的前提是未判定为事件刷怪且未判定为小动物刷怪。

- 如果刷怪点纵坐标在 0.35worldSurface 之上，并且是困难模式，判定为太空刷怪。
- 如果刷怪点纵坐标在 0.35worldSurface 之上，并且刷怪点到世界中心的水平距离大于世界宽度的 0.05 倍，判定为太空刷怪。
- 如果刷怪点纵坐标在 0.45worldSurface 之上，并且是困难模式，有 1/10 概率判定为太空刷怪。

B.3.5 水中刷怪

如果刷怪面上方的两格都有液体，并且刷怪面上方一格是水，判定为水中刷怪。

B.3.6 大理石刷怪与花岗岩刷怪

如果刷怪面是大理石块，判定为大理石刷怪。如果刷怪面是花岗岩块，判定为花岗岩刷怪。刷怪面既不是大理石块也不是花岗岩块，再看玩家脚下的图格<sup>3</sup>；如果这一格为大理石块，判定为大理石刷怪；如果这一格为花岗岩块，判定为花岗岩刷怪。

如果以上条件均未满足，判定较复杂。如果读者看不懂[算法 1](#)，只用记住，玩家附近和刷怪面附近大理石/花岗岩越多，越容易刷对应怪。进入到这一级判定时，可以同时判定为刷大理石怪和花岗岩怪。

B.3.7 蜘蛛巢刷怪和地下沙漠刷怪

蜘蛛巢刷怪和地下沙漠刷怪都是通过背景墙判定的。蜘蛛巢刷怪通过蜘蛛墙判定，地下沙漠刷怪通过沙岩墙、硬化沙墙和它们的三化对应墙判定。这两个刷怪是独立判定的，可以同时判定为蜘蛛巢刷怪和地下沙漠刷怪。

这两个刷怪的前提是玩家不在地牢环境，未判定为事件刷怪。蜘蛛巢刷怪要求刷怪面在 rockLayer 之下，刷怪面距离世界底端大于 200 格。地下沙漠刷怪要求刷怪面在 rockLayer 之上，刷怪面距离世界顶端大于 200 格。

<sup>3</sup>具体为玩家脚下 8 像素处的图格





```

输入: 刷怪面坐标 (x1,y1), 玩家脚下图格坐标 (x2,y2)
输出: 大理石刷怪判定 flag1, 花岗岩刷怪判定 flag2
flag1=flag2=false;
size1 是 20 到 30 的随机整数;
stepX1 是 1 到 3 的随机整数; stepY1 是 1 到 3 的随机整数;
if x1-size1<0 then size1=x1;
if y1-size1<0 then size1=y1;
if x1+size1>=maxTileX then size1=maxTileX-x1-1;
if y1+size1>=maxTileY then size1=maxTileY-y1-1;
for x in x1-size1:stepX1:x1+size1 do
    for y in y1-size1:stepY1:y1+size1 do
        if (x,y) 处的图格为大理石, flag1=true;
        if (x,y) 处的图格为花岗岩, flag2=true;
    end
end
size2 是 30 到 60 的随机整数;
stepX2 是 3 到 6 的随机整数; stepY2 是 3 到 6 的随机整数;
if x2-size2<0 then size2=x2;
if y2-size2<0 then size2=y2;
if x2+size2>=maxTileX then size2=maxTileX-x2-2;
if y2+size2>=maxTileY then size2=maxTileY-y2-2;
for x in x2-size2:stepX2:x2+size2 do
    for y in y2-size2:stepY2:y2+size2 do
        if (x,y) 处的图格为大理石, flag1=true;
        if (x,y) 处的图格为花岗岩, flag2=true;
    end
end

```

算法 1: 大理石/花岗岩刷怪判定算法

有 1/3 概率通过刷怪面附近区域内背景墙判定。以刷怪面为中心, 取一个正方形区域, 其边长是 11 到 29 的一个奇数格。如果这个正方形区域没有超出世界边界并且其中存在一个对应的墙, 判定成功, 否则判定失败。

在另外 2/3 概率中, 直接通过玩家坐标所在格的背景墙判定。

**注** 即使玩家站在对应的背景墙前, 如果刷怪面附近没有对应背景墙, 仍有 1/3 概率判定失败。

## B.4 刷怪失败

前面已经说过, 选取失败后还会再次选取, 一共有 50 次机会。如果 50 次都没有成功, 那么本次刷怪失败。

如果玩家的活跃敌怪数量达到了刷怪量, 刷怪失败。

如果刷怪面碰撞箱与以某个玩家为中心, 宽  $sWidth + 2safeRangeX$ , 高  $sHeight + 2safeRangeY$  的矩形相交, 本次刷怪失败。

如果在地牢环境, 但是刷怪面不是地牢方块或者刷怪面上方一格没有背景墙, 本次刷怪失败。

如果刷怪面上方两格都是液体，并且刷怪面上方一格是蜂蜜，刷怪失败。

## B.5 刷怪种类

刷怪的优先级是四柱 > 太空 > 事件 > 昏迷男子 > 蜘蛛巢 > 地下沙漠 > 巨骨舌鱼 > 血水母 > 嗜血怪 > 海洋 > 沉睡渔夫 > 食人鱼 > 蓝水母 > 水中小动物 > 受缚哥布林 > 受缚巫师 > 小动物 > 地牢 > 陨石 > 撒旦军团 > 霜月 > 南瓜月 > 日食 > 发光蘑菇地 > 腐地蠕虫 > 宝箱怪 > 幻灵 > 弹跳杰克南瓜灯 > 骷髅博士 > 紫胶虫 > 地下蠕虫 > 老鼠 > 蜗牛 > 丛林青蛙 > 困难模式丛林 > 丛林 > 沙尘暴 > 木乃伊 > 地表神圣 > 附魔剑 > 猩红之地 > 腐化之地 > 地表综合 > 地表白天 > 地表夜晚 > 地下 > 地狱 > 洞穴。

每级的名称不唯一对应某些怪，有些怪会同时出现在多级中。

所有判定完成后，如果新生成的怪是蓝史莱姆，有 1/180 概率被替换为粉史莱姆 (ID:-4)。

### B.5.1 四柱

如果玩家在四柱附近，那么就会刷四柱怪。四柱刷怪的优先级是星云柱 > 星旋柱 > 星尘柱 > 日曜柱。

#### B.5.1.1 星云柱

刷怪为星云浮怪、吮脑怪、进化兽、预言帝。这四个怪的刷怪比例为 1:5:3:3。星云浮怪在整个世界中的上限为 2 个，进化兽在整个世界中的上限为 3 个，预言帝在整个世界中的上限为 2 个。这个刷怪比例 & 上限的规则是四柱的特色。举例来说，没有任何刷怪的时候，这四个怪的刷怪概率分别是 1/12、5/12、1/4、1/4；如果已经刷出了两个星云浮怪，那么不会再刷星云浮怪，剩下三个怪的刷怪概率分别是 5/11、3/11、3/11。



图 B.1: 星云柱

#### B.5.1.2 星旋柱

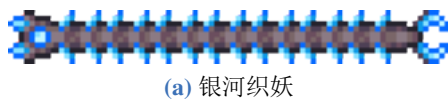
刷怪为淤泥怪、异星蜂王、异星黄蜂、星旋怪。刷怪比例为 2:1:2:4。淤泥怪上限为 3，异星蜂王上限为 3，星旋怪上限为 4。

#### B.5.1.3 星尘柱

刷怪为银河织妖、星细胞、流体入侵怪、闪耀炮手、观星怪。刷怪比例为 1:1:1:2:3。



图 B.2: 星旋柱



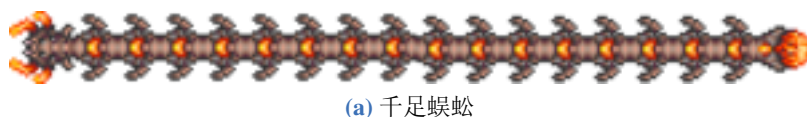
(a) 银河织妖



图 B.3: 星尘柱

### B.5.1.4 日曜柱

刷怪为千足蜈蚣、火龙怪、火龙怪骑士、火滚怪、流星火怪、火月怪、火龙战士。刷怪比例为 1:1:1:1:1:1:1。千足蜈蚣上限为 1，火龙怪上限为 2，火龙怪骑士上限为 1，火龙战士上限为 2。



(a) 千足蜈蚣



图 B.4: 日曜柱

### B.5.2 太空刷怪

优先级是火星飞船 > 火星探测器 > 飞龙 > 鸟妖。

如果执行的是火星入侵的刷怪，那么生成**火星飞船**。

在石巨人后，通过了 Boss 与事件检查，玩家中心在**透光墙**或**云墙**前，刷怪面到世界中心的横坐标距离大于世界宽度  $\times 0.165^4$ ，那么有概率生成**火星探测器**，这个概率与是否打过火星入侵、是否在水蜡烛区域、是否有水蜡烛 buff 相关（表 B.1）。水蜡烛区域和水蜡烛 buff 不是一回事，水蜡烛区域不包括手持水蜡烛的情况。火星探测器的上限为 1。

000	001	011	100	101	111
1/8	15/64	5/9	1/30	59/900	19/100

表 B.1: 二进制的第一位表示是否打过火星入侵，第二位表示是否在水蜡烛区域，第三位表示是否有水蜡烛 buff。例如 101 表示打过火星入侵，不在水蜡烛区域内，有水蜡烛 buff。

没有水蜡烛 buff 的时候，**飞龙**的生成概率为 1/10；有水蜡烛 buff 的时候生成概率为 19/100。飞龙的上限为 1。当禁止穿墙刷怪时不会刷飞龙。

如果火星飞船、火星探测器、飞龙均未生成，那么生成鸟妖。



图 B.5: 太空刷怪

B.5.3 事件刷怪

B.5.3.1 哥布林入侵

1/9 概率生成**哥布林巫士**，8/45 概率生成**哥布林苦力**，32/135 概率生成**哥布林弓箭手**，32/405 概率生成**哥布林盗贼**，64/405 概率生成**哥布林战士**。在困难模式中，有 1/30 概率生成**哥布林召唤师**，这会覆盖前面的生成。哥布林召唤师上限为 1。



图 B.6: 哥布林入侵

<sup>4</sup>小世界为 693 格，中世界为 1056 格，大世界为 1386 格



### B.5.3.2 雪人入侵

1/7 概率生成巴拉雪人，2/7 概率生成雪人暴徒；4/7 概率生成戳刺先生。



图 B.7: 雪人入侵

### B.5.3.3 海盗入侵

1/11 概率生成海盗弩手，10/99 概率生成鹦鹉，80/693 概率生成海盗神射手，160/693 概率生成私船海盗，320/693 概率生成海盗水手。

海盗船长有 1/30 概率生成，会覆盖前面的生成。海盗船长上限为 1。

荷兰飞盗船生成要求入侵进度超过一半，并且刷怪面的左右各 20 格，上方 10 格到 40 格范围内没有实体块。荷兰飞盗船生成概率是 1/20。荷兰飞盗船上限为 1。荷兰飞盗船的生成会覆盖其他生成。



(g) 荷兰飞盗船

图 B.8: 海盗入侵

### B.5.3.4 火星入侵

火星飞碟的生成分为两段判定。离入侵结束不到 100 分<sup>5</sup>时，火星飞碟的概率为 1/10 并且会覆盖其他生成（包括第二段）。第二段中火星飞碟和其他敌怪处理方法相同。

火星飞碟上限为 1，火星走妖上限为 1。以下是第二段判定。

火星飞碟概率为 1/70，鳞甲怪和火星工程师概率均为 9/140（火星飞碟达到上限的话，这个概率变为 1/14），火星飞船概率为 2/35，扰脑怪概率为 4/35，激光枪手概率为 4/35，火星走妖概率为 1/7，灰咕噜兽、电击怪和火星军官概率均为 1/7（火星走妖达到上限的话，这个概率为 4/21）。



图 B.9: 火星入侵

### B.5.4 昏迷男子

满足昏迷男子生成条件，并且不是水中刷怪，那么有 1/80 概率生成昏迷男子。



图 B.10: 昏迷男子

### B.5.5 蜘蛛巢

刷怪面有蜘蛛墙，不高于 rockLayer，距离世界底端大于 210 格，且不是水中刷怪，未解救过发型师，那么有 1/8 概率生成织网发型师。

在未生成织网发型师的前提下，如果刷怪面有蜘蛛墙或者判定为蜘蛛巢刷怪，那么困难模式生成黑隐士，困难模式前生成爬墙蜘蛛。

<sup>5</sup>入侵事件总分为 160+40× 玩家数量



图 B.11: 蜘蛛巢

### B.5.6 地下沙漠

进行地下沙漠刷怪的要求是判定为地下沙漠刷怪，刷怪面在地下，并且刷怪面上的背景墙是硬化沙墙/沙岩墙，或者它们的转化墙<sup>6</sup>。刷怪优先级是沙虫 > 墓穴爬虫 > 困难模式 > 其他。

沙虫和墓穴爬虫的生成都要求不禁止穿墙刷怪，并且刷怪面在地表下 100 格以下。沙虫的概率为 1/33，墓穴爬虫的概率为 1/22。沙虫只会在困难模式生成。

在困难模式中，有 4/5 的概率进行困难模式刷怪。困难模式的刷怪有腐恶食尸鬼、红染食尸鬼、神梦食尸鬼、食尸鬼、沙漠幽魂、邪恶拉弥亚、沙贼、拉弥亚、蛇蜥怪，它们的刷怪比例为 2:2:2:2:1:1:1:1:1，腐恶食尸鬼在腐化环境生成，红染食尸鬼在血腥环境生成，神梦食尸鬼在神圣环境生成，食尸鬼只在纯净环境生成，沙漠幽魂和邪恶拉弥亚在腐化或血腥环境生成，沙贼和拉弥亚在没有腐化和血腥的环境生成，蛇蜥怪不挑环境。举例来说，如果玩家同时处在血腥和神圣环境中，那么可以生成红染食尸鬼、神梦食尸鬼、沙漠幽魂、邪恶拉弥亚、蛇蜥怪，其刷怪比例为 2:2:1:1:1。



(a) 沙虫



(b) 墓穴爬虫



(c) 沙贼



(d) 蛇蜥怪



(e) 食尸鬼



(f) 腐恶食尸鬼



(g) 红染食尸鬼



(h) 神梦食尸鬼



(i) 拉弥亚



(j) 邪恶拉弥亚



(k) 沙漠幽魂



(l) 蚁狮



(m) 蚁狮马



(n) 蚁狮蜂

图 B.12: 地下沙漠

既没刷出沙虫或墓穴爬虫，也没有困难模式刷怪，那么刷蚁狮/蚁狮马/蚁狮蜂，刷怪

<sup>6</sup>神圣化、腐化、血腥化



比例为 1:3:1。

### B.5.7 巨骨舌鱼

困难模式 + 水中刷怪 + 丛林环境，2/3 概率生成巨骨舌鱼。

### B.5.8 血水母

困难模式 + 水中刷怪 + 血腥环境，2/3 概率生成血水母。

### B.5.9 嗜血怪

困难模式 + 水中刷怪 + 血腥环境，2/3 概率生成嗜血怪。



图 B.13

### B.5.10 海洋

要求：水中刷怪，刷怪横坐标距离地图左右边界 <250 格，刷怪面是沙块或珍珠/黑檀/猩红沙块，刷怪面纵坐标在 rockLayer 之上。优先级：沉睡渔夫 > 其他。

#### B.5.10.1 沉睡渔夫

要求：刷怪横坐标在安全区域外，满足沉睡渔夫生成条件。

从刷怪面向上搜索 47 格（不包括刷怪面）找到第一个可以生成沉睡渔夫的图格，这一格需要满足：没有实体块；上方第一格没有实体块；上方第二格没有液体；上方第二格没有实体块。如果找到了这一格，那么生成沉睡渔夫。

#### B.5.10.2 其他

1/60 概率生成海蜗牛，59/1500 概率生成乌贼，59/500 概率生成鲨鱼，413/1500 概率生成螃蟹，413/750 概率生成粉水母。



图 B.14: 海洋

### B.5.11 沉睡渔夫

要求：不是水中刷怪，刷怪横坐标距离地图左右边界 <340 格，刷怪面是沙块或珍珠/黑檀/猩红沙块刷怪面纵坐标在 worldSurface 之上，满足沉睡渔夫生成条件。生成沉睡渔夫。

### B.5.12 食人鱼

要求：水中刷怪。刷怪面在 rockLayer 之下，有 1/2 概率生成食人鱼；刷怪面是丛林草皮，必然生成食人鱼。在困难模式，生成的食人鱼有 2/3 概率转化为琵琶鱼。

### B.5.13 蓝水母

要求：水中刷怪，刷怪面在 worldSurface 之下。有 1/3 概率生成蓝水母。困难模式中生成的蓝水母转化为绿水母。



图 B.15

### B.5.14 水中小动物

如果是水中刷怪，在腐化环境有 1/4 概率生成腐化金鱼。

如果是水中刷怪，不在腐化环境，刷怪面在 worldSurface 之上，刷怪面距离世界顶端大于 50 格，在白天，有 1/6 概率在水面<sup>7</sup>等概率生成鸭或野鸭。上一句话中没有成功生成的话，生成金鱼。

如果未判定生成小动物，那么生成友好水中小动物的概率额外乘 1/4。



图 B.16: 水中小动物

### B.5.15 受缚哥布林

要求：满足受缚哥布林生成条件，不是水中刷怪，刷怪面不在 rockLayer 之上，刷怪面距离世界底端大于 210 格。有 1/20 概率生成受缚哥布林。

<sup>7</sup>这里的水面判定与海洋沉睡渔夫的判定相同。

### B.5.16 受缚巫师

要求：满足受缚巫师生成条件，，不是水中刷怪，刷怪面不在 rockLayer 之上，刷怪面距离世界底端大于 210 格。有 1/20 概率生成受缚巫师。



图 B.17

### B.5.17 小动物

要求：判定生成小动物，不是水中刷怪。丛林草皮上有 1/150 概率生成金蛙，149/150 概率生成青蛙。雪块和冰雪块上企鹅和蓝企鹅生成概率各半。在草皮或神圣草皮上的刷怪优先级：雨天 > 萤火虫 > 鸟 (1) > 蝴蝶 > 鸟 (2) > 兔兔和松鼠。其他情况下，如果刷怪面不在 worldSurface 之下，生成失败；否则生成规则与草皮上的生成规则相同，除了在刷怪面是沙块时，兔兔和松鼠会被各 1/2 概率生成的黑蝎子和蝎子替代。

#### B.5.17.1 雨天

在雨天，2/3 概率生成蠕虫，1/3 概率生成步行金鱼。此外，有 1/150 的概率它们会被金蠕虫覆盖。

#### B.5.17.2 萤火虫

要求：在晚上，刷怪面不在 worldSurface 之下。如果刷怪面是草皮，生成萤火虫；否则生成荧光虫。有 1/fireFlyFriendly 的概率生成。在成功生成的前提下，在刷怪面的上下左右共 4 格分别有 1/fireFlyMultiple 的概率额外生成一个。

有 1/9 的概率，fireFlyFriendly 在 1 到 3 随机，fireFlyMultiple 在 3 到 7 随机；有 8/27 的概率，fireFlyFriendly 和 fireFlyMultiple 均为 999999；有 16/27 的概率，fireFlyFriendly 在 2 到 14 随机，fireFlyMultiple 在 6 到 29 随机。它们的值在每天入夜时刷新。

#### B.5.17.3 鸟 (1)

要求：在 4:30 到 9:30，刷怪面不在 worldSurface 之下。有 2/3 概率生成某种鸟。

在确定生成某种鸟的前提下，有 1/4 概率生成红雀，1/4 概率生成冠蓝鸦，1/2 概率生成鸟。此外，有 1/150 的概率它们会被金鸟覆盖。

#### B.5.17.4 蝴蝶

要求：白天，刷怪面不在 worldSurface 之下。生成某种蝴蝶的概率是 1/butterflyChance。butterflyChance 的值在每天入夜时刷新，有 1/3 概率为 999999，剩下 2/3 概率在 1 到 24

随机。

在确定生成某种蝴蝶的前提下，有 1/150 概率生成**金蝴蝶**，149/150 概率生成**蝴蝶**。此外，在刷怪面的左右共 2 格分别有 1/4 的概率额外生成一个蝴蝶。

### B.5.17.5 鸟 (2)

要求：刷怪面不在 worldSurface 之下。有 1/2 概率生成某种鸟。其他刷怪概率与鸟 (1) 相同。

### B.5.17.6 兔兔和松鼠

各种松鼠要求刷怪面不在 worldSurface 之下。刷怪优先级：金兔 > 金松鼠 > 史莱姆兔兔 > 圣诞节兔兔 > 派对兔兔 > 松鼠 = 红松鼠 > 兔兔。**金兔**概率 1/150，**金松鼠**概率 1/150，**史莱姆兔兔**概率 2/3（要求万圣节期间），**圣诞节兔兔**概率 2/3（要求圣诞节期间），**派对兔兔**概率 2/3（要求派对期间），**松鼠**和**红松鼠**概率各 1/6，以上所有均未成功生成的，生成**兔兔**。



图 B.18: 小动物

### B.5.18 地牢

要求：在地牢环境。刷怪优先级：**地牢守卫**>**受缚机械师**>**骷髅李小龙**>**骷髅突击手**=**骷髅狙击手**=**骷髅特警**>**圣骑士**=**巨型诅咒骷髅头**>**褴褛邪教徒法师**=**死灵法师**=**魔教徒**>**生锈装甲骷髅**=**蓝装甲骷髅**=**地狱装甲骷髅**>**地牢史莱姆**=**尖刺球**=**烈焰火轮**=**诅咒骷髅头**>**暗黑法师**>**愤怒骷髅怪**。部分敌怪只会在击败世纪之花后的困难模式出现，而且部分敌怪的生成依赖于刷怪面的背景墙种类，这两个信息下文会省略。

如果没打过骷髅王，生成**地牢守卫**。未解救**机械师**，刷怪面在 rockLayer 之下，有 1/5 概率生成**受缚机械师**。**骷髅李小龙**概率 1/30。**骷髅突击手**、**骷髅狙击手**、**骷髅特警**概率

都是 1/15。圣骑士概率 1/35，世界中只能存在一个。巨型诅咒骷髅头概率 1/30。褴褛邪教徒法师、死灵法师、魔教徒概率都是 1/20；它们各有两个 ID，生成概率均等；世界中分别只能存在一个。生锈装甲骷髅、蓝装甲骷髅、地狱装甲骷髅概率都是 2/3；它们各有 4 个 ID，生成概率均等。地牢史莱姆概率 1/37。尖刺球概率 1/4；要求以刷怪面为中心的边长 600 像素的正方形与最近的另一个尖刺球的支撑块不相交。烈焰火轮概率 1/15。诅咒骷髅头概率 1/9。暗黑法师概率 1/7。

以上均未生成时，生成愤怒骷髅怪。愤怒骷髅怪一共有 6 个 ID：294、295、296、31、-14、-13，它们的概率分别是 1/5、1/5、1/5、6/25、1/10、3/50。

### B.5.19 陨石

在陨石环境，生成流星头。

### B.5.20 撒旦军队

撒旦军队事件在进行时，并且玩家在事件区域内，那么不刷怪。

### B.5.21 霜月

要求：刷怪面不高于 worldSurface，在晚上，在霜月中。刷怪优先级：礼物宝箱怪>其他。

与波数无关的是礼物宝箱怪。只要世界中只存在不到 4 个礼物宝箱怪，它就会以 1/30 的概率生成。

第 20 波，冰雪女王、圣诞坦克、常绿尖叫怪概率各 1/3。

第 19 波，刷怪优先级：冰雪女王>圣诞坦克>常绿尖叫怪>雪兽。冰雪女王概率 1/10，上限 4 个。圣诞坦克概率 1/10，上限 5 个。常绿尖叫怪概率 1/10，上限 7 个。这三个怪都不生成的情况下生成雪兽。

第 18 波，刷怪优先级：冰雪女王>圣诞坦克>常绿尖叫怪>其他。冰雪女王概率 1/10，上限 3 个。圣诞坦克概率 1/10，上限 4 个。常绿尖叫怪概率 1/10，上限 6 个。这三个怪都不生成的情况下，胡桃夹士概率 1/3，坎卜斯概率 2/9，雪兽概率 4/9。

第 17 波，刷怪优先级：冰雪女王>圣诞坦克>常绿尖叫怪>其他。冰雪女王概率 1/10，上限 2 个。圣诞坦克概率 1/10，上限 3 个。常绿尖叫怪概率 1/10，上限 5 个。这三个怪都不生成的情况下，精灵直升机概率 1/4，坎卜斯概率 3/8，雪兽概率 3/8。

第 16 波，刷怪优先级：冰雪女王>圣诞坦克>常绿尖叫怪>其他。冰雪女王概率 1/10，上限 2 个。圣诞坦克概率 1/10，上限 2 个。常绿尖叫怪概率 1/10，上限 4 个。这三个怪都不生成的情况下，雪花怪概率 1/2，雪兽概率 1/2。

第 15 波，刷怪优先级：冰雪女王>圣诞坦克>常绿尖叫怪>其他。冰雪女王概率 1/10，上限 1 个。圣诞坦克概率 1/10，上限 2 个。常绿尖叫怪概率 1/10，上限 3 个。这三个怪都不生成的情况下，精灵直升机概率 1/3，雪兽概率 2/3。



图 B.19: 地牢



图 B.20: 流星头

第 14 波，刷怪优先级：冰雪女王>圣诞坦克>常绿尖叫怪>其他。冰雪女王概率 1/10，上限 1 个。圣诞坦克概率 1/10，上限 1 个。常绿尖叫怪概率 1/10，上限 1 个。这三个怪都不生成的情况下，雪兽概率 1/3，不刷怪概率 2/3。

第 13 波，刷怪优先级：冰雪女王>圣诞坦克>其他。冰雪女王概率 1/10，上限 1 个。圣诞坦克概率 1/10，上限 1 个。这两个怪都不生成的情况下，雪花怪概率 1/3，雪兽概率 1/9，姜饼人概率 5/27，精灵直升机概率 10/27。

第 12 波，刷怪优先级：冰雪女王>常绿尖叫怪>其他。冰雪女王概率 1/10，上限 1 个。常绿尖叫怪概率 1/10，上限 1 个。这两个怪都不生成的情况下，雪兽概率 1/8，姜饼人概率 7/24，僵尸精灵概率 7/12。僵尸精灵的三个 ID 等概率生成。

第 11 波，刷怪优先级：冰雪女王>其他。冰雪女王概率 1/10，上限 1 个。冰雪女王不生成的情况下，雪花怪概率 1/6，姜饼人概率 5/12，僵尸精灵概率 5/12。

第 10 波，刷怪优先级：圣诞坦克>常绿尖叫怪>其他。圣诞坦克概率 1/10，上限 1 个。常绿尖叫怪概率 1/10，上限 2 个。这两个怪都不生成的情况下，坎卜斯概率 1/6，胡桃夹士概率 5/18，精灵直升机概率 5/27，僵尸精灵概率 10/27。

第 9 波，刷怪优先级：圣诞坦克>常绿尖叫怪>其他。圣诞坦克概率 1/10，上限 1 个。常绿尖叫怪概率 1/10，上限 1 个。这两个怪都不生成的情况下，胡桃夹士概率 1/2，精灵直升机概率 1/6，姜饼人概率 1/3。

第 8 波，刷怪优先级：圣诞坦克>其他。圣诞坦克概率 1/10，上限 1 个。圣诞坦克不生成的情况下，坎卜斯概率 1/8，胡桃夹士概率 7/24，精灵直升机概率 7/36，精灵弓箭手概率 7/18。

第 7 波，刷怪优先级：圣诞坦克>其他。圣诞坦克概率 1/10，上限 1 个。圣诞坦克不生成的情况下，姜饼人概率 1/3，精灵弓箭手概率 1/6，僵尸精灵概率 1/2。

第 6 波，刷怪优先级：常绿尖叫怪>其他。常绿尖叫怪概率 1/10，上限 2 个。常绿尖叫怪不生成的情况下，精灵直升机概率 1/4，wiki 胡桃夹士概率 3/8，精灵弓箭手概率 3/8。

第 5 波，刷怪优先级：常绿尖叫怪>其他。常绿尖叫怪概率 1/10，上限 1 个。常绿尖叫怪不生成的情况下，精灵弓箭手概率 1/4，wiki 胡桃夹士概率 3/32，僵尸精灵概率 21/32。

第 4 波，刷怪优先级：常绿尖叫怪>其他。常绿尖叫怪概率 1/10，上限 1 个。常绿尖叫怪不生成的情况下，精灵弓箭手概率 1/4，姜饼人概率 1/4，僵尸精灵概率 1/2。

第 3 波，胡桃夹士概率 1/8，精灵弓箭手概率 7/32，姜饼人概率 7/32，僵尸精灵概率 7/16。

第 2 波，精灵弓箭手概率 1/3，僵尸精灵概率 2/3。

第 1 波，姜饼人概率 1/3，僵尸精灵概率 2/3。

### B.5.22 南瓜月

要求：刷怪面不高于 worldSurface，在晚上，在南瓜月中。





(a) 冰雪女王



(b) 圣诞坦克



(c) 常绿尖叫怪



(d) 雪兽



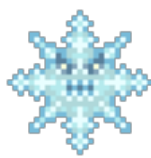
(e) 胡桃夹士



(f) 坎卜斯



(g) 精灵直升机



(h) 雪花怪



(i) 姜饼人



(j) 僵尸精灵



(k) 精灵弓箭手



(l) 礼物宝箱怪

图 B.21: 霜月

第 1 波，生成稻草人，10 个 ID 等概率生成。

第 2 波，树精概率 1/3，稻草人概率 2/3。

第 3 波，地狱犬概率 1/6，树精概率 5/18，稻草人概率 5/9。

第 4 波，刷怪优先级：哀木 > 其他。哀木概率 1/10，上限 1 个。哀木不生成的情况下，地狱犬概率 1/10，树精概率 9/20，稻草人概率 9/20。

第 5 波，刷怪优先级：哀木 > 其他。哀木概率 1/10，上限 1 个。哀木不生成的情况下，胡闹鬼概率 1/8，地狱犬概率 7/40，树精概率 7/20，稻草人概率 7/20。

第 6 波，刷怪优先级：哀木 > 其他。哀木概率 1/7，上限 2 个。哀木不生成的情况下，胡闹鬼概率 1/6，地狱犬概率 5/18，树精概率 5/9。

第 7 波，刷怪优先级：南瓜王 > 其他。南瓜王概率 1/10，上限 1 个。南瓜王不生成的情况下，胡闹鬼概率 1/8，地狱犬概率 7/40，稻草人概率 7/10。

第 8 波，刷怪优先级：南瓜王 > 其他。南瓜王概率 1/10，上限 1 个。南瓜王不生成的情况下，胡闹鬼概率 1/5，地狱犬概率 4/15，树精概率 8/15。

第 9 波，刷怪优先级：南瓜王 > 哀木 > 无头骑士 > 其他。南瓜王概率 1/8，上限 1 个。哀木概率 1/8，上限 1 个。无头骑士概率 1/10，上限 1 个。这三个怪都不生成的情况下生成稻草人。

第 10 波，刷怪优先级：南瓜王 > 哀木 > 无头骑士 > 其他。南瓜王概率 1/10，上限 1 个。哀木概率 1/10，上限 1 个。无头骑士概率 1/10，上限 1 个。这三个怪都不生成的情况下，胡闹鬼概率 1/8，地狱犬概率 7/40，树精概率 7/10。

第 11 波，刷怪优先级：哀木 > 无头骑士 > 其他。哀木概率 1/7，上限 2 个。无头骑士概率 1/10，上限 1 个。这两个怪都不生成的情况下，胡闹鬼概率 1/10，地狱犬概率 9/70，树精概率 9/35，稻草人概率 18/35。如果世界中没有南瓜王，还有 1/10 概率额外生成一个南瓜王。

第 12 波，刷怪优先级：哀木 > 无头骑士 > 其他。哀木概率 1/7，上限 2 个。无头骑士概率 1/7，上限 2 个。这两个怪都不生成的情况下，胡闹鬼概率 1/7，地狱犬概率 6/35，树精概率 24/35。如果世界中南瓜王数量小于 2，还有 1/7 概率额外生成一个南瓜王。

第 13 波，刷怪优先级：哀木 > 无头骑士 > 其他。哀木概率 1/5，上限 3 个。无头骑士概率 1/5，上限 3 个。这两个怪都不生成的情况下，胡闹鬼概率 1/3，地狱犬概率 2/3。如果世界中南瓜王数量小于 2，还有 1/7 概率额外生成一个南瓜王。

第 14 波，刷怪优先级：南瓜王 > 哀木 > 其他。南瓜王概率 1/5，上限 3 个。哀木概率 1/5，上限 3 个。这两个怪都不生成的情况下生成无头骑士。

第 15 波，南瓜王和哀木概率各为 1/2。

### B.5.23 日食

要求：刷怪面不在 worldSurface 之下，在白天，在日食中。刷怪优先级：蛾怪 > 眼怪 > 变态人 > 钉头 > 致命球 > 吸血鬼 > 死神 > 攀爬魔 > 飞人博士 > 屠夫 > 科学怪人 = 水月怪 = 弗里茨 = 沼泽怪。



蛾怪 (ID:477) 概率 1/80, 要求击败三个机械 Boss, 上限 1 个。眼怪 (ID:251) 概率 1/50, 上限 1 个。变态人 (ID:466) 概率 1/5, 要求击败世纪之花, 上限 1 个。钉头 (ID:463) 概率 1/20, 要求击败世纪之花, 上限 1 个。致命球 (ID:467) 概率 1/20, 要求击败世纪之花, 上限 2 个。吸血鬼 (ID:159) 概率 1/15。死神 (ID:253) 概率 1/13, 要求击败三个机械 Boss。攀爬魔 (ID:469) 概率 1/8。飞人博士 (ID:468) 概率 1/7, 要求击败世纪之花。屠夫 (ID:460) 概率 1/5, 要求击败世纪之花。科学怪人 (ID:162)、水月怪 (ID:461)、弗里茨 (ID:462)、沼泽怪 (ID:166) 概率各 1/4。

#### B.5.24 发光蘑菇地

要求：刷怪面为蘑菇草皮。刷怪优先级：水中刷怪 > 地表刷怪 > 地下刷怪。地表刷怪和地下刷怪由刷怪面位置决定，它们的交界为 worldSurface。需要注意的是，刷怪面在 worldSurface 上时，地表刷怪和地下刷怪都会有效，此时应用优先级关系。

水中刷怪，困难模式会刷蘑菇鱼 (ID:256)。

地表刷怪，有 2/3 概率成功刷怪。在这部分概率里，刷怪优先级为发光蜗牛 > 蘑菇僵尸 = 孢子僵尸 = 歪尾真菌 = 瓢虫 > 巨型真菌球怪 > 真菌球怪。发光蜗牛 (ID:360) 在困难模式概率 1/12，困难模式之前概率 37/72。蘑菇僵尸 (ID:255) 和孢子僵尸 (ID:254) 概率各 1/3，歪尾真菌 (ID:257) 和瓢虫 (ID:258) 概率各 1/8。巨型真菌球怪 (ID:260) 概率 2/3，要求困难模式。以上所有敌怪均未生成，则生成真菌球怪 (ID:259)。

地下刷怪要求困难模式，有 2/3 概率成功刷怪。在这部分概率里，刷怪优先级为松露虫 > 发光蜗牛 > 歪尾真菌 = 瓢虫 > 巨型真菌球怪 > 真菌球怪。松露虫 (ID:374) 概率为 1/5，要求困难模式。发光蜗牛在困难模式概率 1/8，困难模式之前概率 11/32。歪尾真菌和瓢虫概率各 1/8。巨型真菌球怪 (ID:260) 概率 2/3，要求困难模式。以上所有敌怪均未生成，则生成真菌球怪 (ID:259)。

#### B.5.25 腐地蠕虫

要求：腐化环境，禁止穿墙刷怪。进行腐地蠕虫刷怪的概率是 1/65。困难模式之前生成吞噬怪 (ID:7)。困难模式中吞噬怪概率 1/4，吞世怪 (ID:98) 概率 3/4。

#### B.5.26 宝箱怪

第一轮判定要求：困难模式，刷怪面在 worldSurface 之下。刷宝箱怪的概率是 1/75。

刷怪优先级：腐化宝箱怪 > 猩红宝箱怪 > 神圣宝箱怪 > 普通宝箱怪。腐化宝箱 (ID:473) 怪概率 1/2，要求腐化环境，上限为 1 个。猩红宝箱怪 (ID:474) 概率 1/2，要求血腥环境，上限为 1 个。神圣宝箱怪 (ID:475) 概率 1/2，要求神圣环境，上限为 1 个。这三个环境宝箱怪都不生成，则生成普通宝箱怪 (ID:85)。

在第一轮判定失败的情况下，进行第二轮判定。第二轮判定要求：困难模式，刷怪面上方一格是天然土墙。有 1/20 概率刷普通宝箱怪。

**注** 如果环境为腐化、猩红、神圣的混合环境，并且符合第一轮判定条件，不符合第二轮

判定条件，那么生成的第一个宝箱怪，有 1/2 概率是腐化宝箱怪，1/4 概率是猩红宝箱怪，1/8 概率是神圣宝箱怪，1/8 概率是普通宝箱怪。

### B.5.27 幻灵 (ID:82)

要求：困难模式，刷怪面不低于 worldSurface，在晚上。新月期间概率 6/25，其他时间为 1/20。

### B.5.28 弹跳杰克南瓜灯 (ID:304)

要求：困难模式，万圣节期间，刷怪面不低于 worldSurface，在晚上。概率 1/10。

### B.5.29 骷髅博士 (ID:52)

要求：刷怪面为丛林草皮，在晚上。概率 1/500。

### B.5.30 紫胶虫 (ID:219)

要求：刷怪面为丛林草皮，刷怪面低于 worldSurface。概率 1/60。

### B.5.31 地下蠕虫

要求：刷怪面低于 worldSurface，刷怪面距离世界底端大于 210 格，不在苔原环境，不在血腥环境，不在腐化环境，不在丛林环境，不在神圣环境。概率 1/8。生成的蠕虫有 1/150 概率被替换为金蠕虫。

### B.5.32 老鼠

要求：刷怪面低于 worldSurface，刷怪面距离世界底端大于 210 格，不在苔原环境，不在血腥环境，不在腐化环境，不在丛林环境，不在神圣环境。概率 1/13。生成的老鼠 (ID:300) 有 1/150 概率被替换为金老鼠 (ID:447)。

### B.5.33 蜗牛 (ID:359)

要求：刷怪面低于 worldSurface，刷怪面高于 rockLayer 与世界底端的中点，不在苔原环境，不在血腥环境，不在腐化环境，不在神圣环境。概率 1/13。

### B.5.34 丛林青蛙

要求：刷怪面高于 worldSurface，丛林环境。青蛙概率 1/9。生成的青蛙有 1/150 概率被替换为金蛙。

B.5.35 困难模式丛林

要求：刷怪面为丛林草皮，困难模式。进行困难模式丛林刷怪的概率为 2/3。刷怪优先级：地表丛林 = 地下丛林 > 其他

B.5.35.1 地表丛林

要求：刷怪面高于 worldSurface。在晚上有 1/3 概率生成巨型飞狐 (ID:152)，在白天有 3/4 概率生成跳跳兽 (ID:177)。

B.5.35.2 地下丛林

要求：刷怪面低于 worldSurface。蛾 (ID:205) 概率 1/100，丛林蜘蛛 (ID:236) 概率 99/500，青苔黄蜂 (ID:176/-18/-19/-20/-21) 概率 297/1000。青苔黄蜂的 5 个 ID 生成比例为 6561:729:810:900:1000。

B.5.35.3 其他

愤怒捕手 (ID:175) 概率 1/3，巨型陆龟 (ID:153) 概率 2/3。  
**注** 如果刷怪面恰好在 worldSurface 上，那么地表丛林和地下丛林都会跳过，只会生成愤怒捕手和巨型陆龟。

B.5.36 丛林

丛林的三种刷怪环境是互斥的，所以不存在优先级问题。神庙和地下丛林一定会刷怪，地表丛林不一定。

B.5.36.1 神庙

要求：玩家中心在天然丛林蜥蜴砖墙前，刷怪面是丛林蜥蜴砖。丛林蜥蜴 (ID:198) 概率 2/3，飞蛇 (ID:226) 概率 1/3。

B.5.36.2 地下丛林

要求：刷怪面是丛林草皮，刷怪面在 worldSurface 和 rockLayer 的中点之下。  
尖刺丛林史莱姆 (ID:204) 概率 1/4，食人怪 (ID:43) 概率 3/16，黄蜂 (ID:42/231/232/233/234/235/-16/-17/-56/-57/-58/-59/-60/-61/-62/-63/-64/-65) 概率 9/16。这所有 ID 对应的黄蜂变种见表 B.2。六种主黄蜂变种的刷怪比例为 3:1:1:1:1:1，原变种、小变种和大变种的刷怪比例为 9:3:4。**黄蜂 Wiki**

	黄蜂	肥胖黄蜂	蜂蜜黄蜂	多叶黄蜂	尖刺黄蜂	毒刺黄蜂
原变种	42	231	232	233	234	235
小变种	-16	-56	-58	-60	-62	-64
大变种	-17	-57	-59	-61	-63	-65

表 B.2: 所有黄蜂变种的 ID





### B.5.36.3 地表丛林

要求：刷怪面是丛林草皮，刷怪面不在 worldSurface 和 rockLayer 的中点之下。丛林蝙蝠 (ID:51) 概率 1/4，抓人草 (ID:56) 概率 3/32。

### B.5.37 沙尘暴

要求：正在进行沙尘暴，在沙尘暴区域中，刷怪面是沙块/黑檀沙块/猩红沙块/珍珠沙块，刷怪面通过了沙块的局部聚簇检查。

未击杀过克苏鲁之眼，困难模式前。愤怒翻滚怪 (ID:546) 概率 1/2，蚁狮马概率 1/4，蚁狮 (ID:69) 概率 1/8，秃鹰 (ID:61) 概率 1/8。

其他情况。沙尘精 (ID:541) 概率 1/20，上限 1 个，要求困难模式。沙虫 (ID:510) 概率 1/3，上限 4 个，要求困难模式，禁止穿墙刷怪。沙鲨概率 1/2，要求困难模式，禁止穿墙刷怪；根据刷怪面的不同，沙块上生成沙鲨 (ID:542)，黑檀沙块上生成噬骨沙鲨 (ID:543)，猩红沙块上生成戮血沙鲨 (ID:544)，珍珠沙块上生成晶狐沙鲨 (ID:545)。木乃伊概率 1/3，要求困难模式；根据刷怪面的不同，沙块上生成木乃伊 (ID:78)，黑檀沙块或猩红沙块上生成暗黑木乃伊 (ID:79)，珍珠沙块上生成光明木乃伊 (ID:80)。以上均未成功生成，愤怒翻滚怪概率 1/2，蚁狮马概率 1/4，蚁狮蜂概率 1/4。

### B.5.38 木乃伊

要求：困难模式及某些特定刷怪面。沙块上普通木乃伊概率为 1/3。黑檀沙块或猩红沙块上暗黑木乃伊概率为 1/2。珍珠沙块上光明木乃伊概率 1/2。

### B.5.39 地表神圣

要求：困难模式，不是水中刷怪，刷怪面是珍珠沙块、珍珠石块、神圣草皮、粉冰雪块，刷怪面在 rockLayer 之上。刷怪优先级：腹足怪 > 妖精 = 独角兽。腹足怪 (ID:122) 概率 1/2，要求在晚上。腹足怪没有生成的前提下，水蜡烛区域内妖精 (ID:75) 概率 81/100，独角兽 (ID:86) 概率 19/100；水蜡烛区域外妖精概率 9/10，独角兽概率 1/10。

### B.5.40 附魔剑 (ID:84)

要求：困难模式，不是水中刷怪，禁止穿墙刷怪，刷怪面是珍珠沙块、珍珠石块、神圣草皮、粉冰雪块，刷怪面不在 rockLayer 之上。概率 1/50。

### B.5.41 猩红之地

要求：刷怪面是猩红石块、猩红沙块、红冰雪块、猩红草皮。如果玩家在猩红之地，那么猩红矿也可以做刷怪面。刷怪优先级：恶心浮游怪 > 灵液黏黏怪 > 猩红史莱姆 > 猩红斧 > 蹦蹦兽 > 血爬虫 > 脸怪 = 猩红喀迈拉。

恶心浮游怪 (ID:182) 概率 1/5，要求困难模式，刷怪面不在 rockLayer 之上，禁止穿墙刷怪。灵液黏黏怪 (ID:268) 概率 1/2，要求困难模式，刷怪面不在 rockLayer 之上。猩



红史莱姆 (ID:183/-24/-25) 概率 1/3, 要求困难模式; 3 个 ID 的概率比为 4:2:3。猩红斧 (ID:179) 概率 1/40, 要求困难模式, 刷怪面不在 rockLayer 之上, 禁止穿墙刷怪。蹦蹦兽 (ID:174) 概率 1/2 (刷怪面不在 worldSurface 之下) 或 1 (刷怪面在 worldSurface 之下), 要求困难模式。血爬虫 (ID:239) 概率 3/4 (刷怪面有背景墙) 或 1/8 (刷怪面无背景墙)。脸怪 (ID:181) 和猩红喀迈拉 (ID:173/-22/-23) 概率各为 1/2; 猩红喀迈拉的三个 ID 比例为 4:2:3。

### B.5.42 腐化之地

要求: 刷怪面是黑檀石块、黑檀沙块、紫冰雪块、腐化草皮。如果玩家在腐化之地, 那么魔矿也可以做刷怪面。刷怪优先级:

爬藤怪 (ID:101) 概率 1/3, 要求困难模式, 刷怪面不在 rockLayer 之上。腐化史莱姆 (ID:81) 概率 2/9, 恶翅史莱姆 (ID:121) 概率 1/9, 要求困难模式。诅咒锤 (ID:83) 概率 1/40, 要求困难模式, 刷怪面不在 rockLayer 之上, 禁止穿墙刷怪。腐化者 (ID:94) 概率 1/2 (刷怪面不在 rockLayer 之下) 或 1 (刷怪面在 rockLayer 之下), 要求困难模式。以上所有敌怪均未生成, 则生成噬魂怪 (ID:6/-11/-12), 三个 ID 比例为 4:2:3。

### B.5.43 地表综合

要求: 刷怪面不低于 worldSurface。刷怪优先级: 冰雪巨人 > 彩虹史莱姆 > 愤怒雨云怪 > 火星探测器。

冰雪巨人 (ID:243) 概率 1/20, 上限 1 个; 要求苔原环境, 困难模式, cloudAlpha 大于 0。彩虹史莱姆 (ID:244) 概率 1/20, 上限 1 个; 要求神圣环境, 困难模式, cloudAlpha 大于 0。愤怒雨云怪 (ID:250) 概率 1/10, 上限 2 个; 要求非苔原环境, 困难模式, cloudAlpha 大于 0。火星探测器概率 1/400 (已打过火星入侵) 或 1/100 (未打过火星入侵), 上限 1 个; 要求刷怪面和世界中心横坐标距离大于世界宽度  $\times 0.165$ , 困难模式, 击败过石巨人。

### B.5.44 地表白天

要求: 刷怪面不低于 worldSurface, 在白天。刷怪优先级: 小动物 > 史莱姆王 > 沙漠 > 哥布林侦察兵 > 雨天 > 史莱姆。

#### B.5.44.1 小动物

要求: 刷怪面到重生点的水平距离小于世界宽度的 1/3。刷怪优先级: 企鹅 = 蝴蝶 = 蝎子 > 鸟。

企鹅和蓝企鹅概率相等, 共计 1/15, 要求刷怪面是雪块或冰雪块。蝴蝶刷怪要求刷怪面是草皮或神圣草皮, 刷怪面不在 worldSurface 之下, 概率同 [subsection B.5.17.4](#)。兔兔和松鼠对刷怪面位置没有要求, 概率同 [subsection B.5.17.6](#)。蝎子刷怪要求刷怪面是沙块, 蝎子和黑蝎子概率相等, 共计 1/15。鸟刷怪分两次判定, 有一次成功即可: 第一次判定要求刷怪面是草皮或神圣草皮, 所有鸟总上限 6 个, 时间在 4:30 到 9:30, 进行刷

怪概率为 1/15；第二次判定要求刷怪面是草皮、神圣草皮或雪块，进行刷怪概率为 1/15；各种鸟刷怪比例同subsubsection B.5.17.3。

#### B.5.44.2 史莱姆王 (ID:50)

要求：刷怪面到重生点水平距离大于世界宽度的 1/3，刷怪面是草皮。概率 1/300，上限 1 个。生成位置重新选择，不一定在刷怪面上。

#### B.5.44.3 沙漠

要求：刷怪面是沙块，不是水中刷怪。刷怪优先级：蚁狮 > 沙史莱姆 = 秃鹰。

蚁狮概率 1/5，要求通过沙块局部聚簇检查。沙史莱姆 (ID:537) 概率 1/3，秃鹰概率 2/3。

#### B.5.44.4 哥布林侦察兵

要求：刷怪面到重生点水平距离大于世界宽度的 1/3。概率 1/5（未击败哥布林入侵，敲过暗影珠或猩红之心）或 1/15（其他情况）。

#### B.5.44.5 雨天

要求：在下雨。飞鱼 (ID:224) 概率 1/3，雨伞史莱姆 (ID:225) 概率 1/3。

#### B.5.44.6 史莱姆

刷怪优先级：丛林史莱姆 = 冰雪史莱姆 > 兔兔史莱姆 > 礼物史莱姆 > 绿史莱姆 > 紫史莱姆 > 蓝史莱姆。

刷怪面是丛林草皮，生成丛林史莱姆 (ID:-10)。刷怪面是雪块或冰雪块，生成冰雪史莱姆 (ID:147)。兔兔史莱姆 (ID:302) 概率 2/3，要求万圣节。礼物史莱姆 (ID:333/334/335/336) 概率 2/3，要求圣诞节；4 个 ID 等概率生成。绿史莱姆 (ID:-3) 概率 1/3，要求刷怪面到重生点的水平距离小于 200 格，普通模式。紫史莱姆 (ID:-7) 概率 1/10，要求刷怪面到重生点的水平距离大于 400 格（专家模式没有这个要求）。以上所有史莱姆均未生成，生成蓝史莱姆 (ID:1)。

#### B.5.45 地表夜晚

要求：刷怪面不低于 worldSurface，在晚上。刷怪优先级：萤火虫 > 乌鸦 > 眼球怪 > 小丑 > 僵尸新郎 > 新娘 > 狼人 > 血腥僵尸 = 滴滴怪 > 苔原 > 雨衣僵尸 > 僵尸。

萤火虫生成规则同subsubsection B.5.17.2。乌鸦 (ID:301) 概率 1/10，要求万圣节。眼球怪概率 7/12（新月）或 1/6（其他）。小丑 (ID:109) 概率 1/50，上限 1 个，要求困难模式，血月。僵尸新郎 (ID:53) 概率 1/250，要求血月。新娘 (ID:536) 概率 1/250，要求血月。狼人 (ID:104) 概率 2/3，要求困难模式，满月。装甲幻影魔 (ID:140) 概率 1/3，要求困难模式。血腥僵尸 (ID:489) 和滴滴怪 (ID:490) 概率各 1/5，要求血月。苔原刷怪要求刷怪面

是雪块/冰雪块/紫冰雪块/粉冰雪块。雨衣僵尸 (ID:223/-54/-55) 概率 1/2，要求下雨；三个 ID 的比例为 4:1:1。以上均未生成，生成僵尸。

B.5.45.1 眼球怪

刷怪优先级：游荡眼球怪 > 猫头鹰眼 = UFO 眼 > 恶魔眼。

游荡眼球怪 (ID:133) 概率 1/3，要求困难模式。猫头鹰眼 (ID:317) 和 UFO 眼 (ID:318) 概率各 1/4，要求万圣节。以上均未生成，生成恶魔眼 (ID:2/-43/190/-38/191/-39/192/-40/193/-41/194/-42)。这所有 ID 对应的恶魔眼变种见表 B.3。六种主变种的刷怪比例为 5:1:1:1:1:1，原型和变种的刷怪比例为 2:1。

	恶魔眼	白内障眼	瞌睡眼	胀大眼	绿眼	紫眼
原型	2	190	191	192	193	194
变种	-43	-38	-39	-40	-41	-42

表 B.3: 所有恶魔眼变种的 ID

B.5.45.2 苔原

刷怪优先级：冰雪精 = 狼 > 爱斯基摩僵尸 = 武装爱斯基摩僵尸。

冰雪精 (ID:169) 和狼 (ID:155) 概率各 1/4，要求困难模式。以上未生成，生成爱斯基摩僵尸 (ID:161)；在专家模式，有一半的爱斯基摩僵尸会被替换为武装爱斯基摩僵尸 (ID:431)。

B.5.45.3 僵尸

护士僵尸 (ID:319)、超级英雄僵尸 (ID:320)、妖精僵尸 (ID:321) 概率各 1/6，要求万圣节。圣诞僵尸 (ID:331)、毛衣僵尸 (ID:332) 概率各 1/4，要求圣诞节。其他僵尸分为 7 个主变种，每个主变种下有三个小变种，它们的 ID 见表 B.4。所有主变种的概率都是 1/7，三个小变种的刷怪比例为 4:1:1。在专家模式，除秃头僵尸以外，其余僵尸变种都有 1/3 概率转化为对应的武装变种 (ID:430/432/433/434/435/436)。

	僵尸	秃头僵尸	中箭僵尸	史莱姆僵尸	沼泽僵尸	纤瘦僵尸	女性僵尸
原变种	3	132	186	187	188	189	200
小变种	-26	-28	-30	-32	-34	-36	-44
大变种	-27	-29	-31	-33	-35	-37	-45

表 B.4: 所有僵尸变种的 ID

B.5.46 地下

要求：刷怪面不在 rockLayer 之下。刷怪优先级：巨型蠕虫 > 装甲幻影魔 > 毒泥 > 冰雪史莱姆 > 黄史莱姆 = 蓝史莱姆 = 红史莱姆。

巨型蠕虫 (ID:10) 概率 1/50，要求不在苔原环境，禁止穿墙刷怪；困难模式中巨型蠕虫会被替换为挖掘怪 (ID:95)。装甲幻影魔概率 1/3，毒泥 (ID:141) 概率 1/2，要求困难模



式。如果刷怪面是雪块或冰雪块，或者在苔原环境，生成冰雪史莱姆 (ID:147)。以上均未生成，黄史莱姆 (ID:-9) 概率 1/5，蓝史莱姆概率 2/5，红史莱姆 (ID:-8) 概率 2/5。

### B.5.47 地狱

要求：刷怪面距离世界底端小于 190 格。刷怪优先级：痛苦亡魂 > 骨蛇 > 火焰小鬼 > 其他。

痛苦亡魂 (ID:534) 概率 1/20，上限 1 个，要求困难模式，未解救税收官。骨蛇 (ID:39) 概率 1/40，上限 1 个。火焰小鬼 (ID:24) 概率 1/14。巫毒恶魔 (ID:66) 概率 1/49，恶魔 (ID:62) 概率 6/49，熔岩史莱姆 (ID:59) 概率 2/7，地狱蝙蝠 (ID:60) 概率 4/7；在困难模式，击败一个机械 Boss 后，恶魔有 4/5 概率被替换为红魔鬼 (ID:156)，地狱蝙蝠有 4/5 概率被替换为熔岩蝙蝠 (ID:151)。

### B.5.48 洞穴

刷怪优先级：胭脂虫 > 混沌精 > 猪龙 > 冰雪陆龟 > 巨型蠕虫 > 小雪怪 (1) > 史莱姆之母 > 黑史莱姆 > 蝙蝠 > 骷髅商人 > 迷失女孩 > 符文巫师 > 蒂姆 > 大理石刷怪 > 花岗岩刷怪 > 困难模式常见怪 > 不死矿工 > 亡灵维京海盗 > 小雪怪 (2) > 特殊洞穴怪 > 万圣节骷髅 > 专家模式骷髅 > 普通模式骷髅。

胭脂虫 (ID:217) 概率 1/60；苔原环境中胭脂虫会被替换为青壳虫 (ID:218)。混沌精 (ID:120) 概率 1/8，要求刷怪面是珍珠沙块、珍珠石块、粉冰雪块，困难模式，禁止穿墙刷怪。三种猪龙概率是相互独立的 1/30，优先级腐化猪龙 (ID:170) > 神圣猪龙 (ID:171) > 猩红猪龙 (ID:180)；要求刷怪面是雪块、冰雪块、薄冰、粉冰雪块、紫冰雪块，困难模式，禁止穿墙刷怪；腐化猪龙要求腐化环境，神圣猪龙要求神圣环境，猩红猪龙要求血腥环境。冰雪陆龟 (ID:154) 概率 1/10，要求困难模式，苔原环境。巨型蠕虫概率 1/100，要求禁止穿墙刷怪；困难模式中巨型蠕虫会被替换为挖掘怪；困难模式前的苔原环境中，巨型蠕虫会被替换为小雪怪 (ID:185)。小雪怪概率 1/20，要求苔原环境。史莱姆之母 (ID:16) 概率 1/10，要求困难模式前；在苔原环境，取消史莱姆之母的生成，把第 201 个 NPC 改为尖刺冰雪史莱姆 (ID:184)<sup>8</sup>。黑史莱姆 (ID:-6) 概率 1/4，要求困难模式前；在丛林环境，黑史莱姆会被替换为丛林史莱姆；在苔原环境，黑史莱姆会被替换为尖刺冰雪史莱姆。蝙蝠概率 1/2。骷髅商人 (ID:453) 概率 1/35，上限 1 个。迷失女孩 (ID:195) 概率 1/200（困难模式）或 279/16000（困难模式前）。符文巫师 (ID:172) 概率 1/300，要求困难模式，刷怪面低于 rockLayer 和世界底端的中点。蒂姆 (ID:45) 概率 1/200，要求刷怪面低于 rockLayer 和世界底端的中点；如果玩家装备了任意宝石长袍并且未装备巫师帽，概率会增加到 249/10000。装甲步兵 (ID:481) 概率 3/4，要求大理石刷怪；在困难模式，如果世界中不存在蛇发女妖 (ID:480)，生成的装甲步兵有 5/6 概率被替换为蛇发女妖。花岗岩巨人 (ID:482) 概率 4/5，要求花岗岩刷怪；如果世界中不存在花岗精 (ID:483)，生成的花岗岩巨人有 5/6 概率被替换为花岗精。困难模式常见怪概率 9/10。不死矿工 (ID:44) 概率 1/20。如果刷怪面是雪块、冰雪块、薄冰，生成亡灵维京海盗 (ID:167)。如果在苔原环境，生成小雪怪 (ID:185)。特

<sup>8</sup>源码如此。NPC 数组实际上有 201 个位置

**殊洞穴怪**概率 1/3。新郎骷髅 (ID:322)、宇航员骷髅 (ID:323)、火星骷髅 (ID:324) 概率各 1/6，要求万圣节。专家模式骷髅 (ID:449/452) 概率 1/3，要求专家模式；2 个 ID 比例为 1:3。以上均未生成，生成**普通模式骷髅**变种。

B.5.48.1 蝙蝠

刷怪优先级：夜明史莱姆 > 丛林蝙蝠 > 夜明蝙蝠 > 巨型蝙蝠 > 洞穴蝙蝠 = 冰雪蝙蝠 = 冰雪精。

夜明史莱姆 (ID:138) 概率 1/2，要求困难模式，神圣环境。丛林环境中生成丛林蝙蝠 (ID:51)。困难模式神圣环境中生成夜明蝙蝠 (ID:137)。巨型蝙蝠 (ID:93) 概率 5/6，要求困难模式。如果刷怪面不是雪块、冰雪块、薄冰，生成洞穴蝙蝠 (ID:49)。如果刷怪面是雪块、冰雪块、薄冰，困难模式前生成冰雪蝙蝠 (ID:150)，困难模式生成冰雪精 (ID:169)。

B.5.48.2 困难模式常见怪

要求：困难模式。

苔原环境中，冰雪人鱼 (ID:206) 和装甲维京海盗 (ID:197) 概率各 1/2。

其他环境中，骷髅弓箭手 (ID:110) 和装甲骷髅 (ID:77) 概率各 1/2。如果刷怪面低于 rockLayer 和世界底端的中点，生成的装甲骷髅有 1/5 概率被替换为 ID 为-15 的变种 (Wiki)。在万圣节期间，装甲骷髅及其变种还有 1/5 概率被替换为鬼魂 (ID:316)。

B.5.48.3 特殊洞穴怪

在每个世界中，以 worldID 作为随机数种子，随机选择龙虾、蝶螈、巨型卷壳怪中两种，每种中随机选出 3 个 ID（可重复）。生成龙虾/蝶螈/巨型卷壳怪时，会在 6 个 ID 中随机选择。

B.5.48.4 普通模式骷髅

普通模式骷髅有 4 个主变种，每个主变种下有 3 个小变种。所有主变种概率相等，三个小变种比例为 4:1:1。

	骷髅	头痛骷髅	畸形骷髅	无裤骷髅
原型	21	201	202	203
小	-46	-48	-50	-52
大	-47	-49	-51	-53

表 B.5: 所有普通模式骷髅变种的 ID

B.6 史莱姆雨刷怪

进行史莱姆雨刷怪，要求玩家纵坐标不在 worldSurface+600 像素之下，玩家的活跃敌怪数量超过了 15。进行刷怪的基础概率是 1/(45+30× 玩家的活跃敌怪数量)；在专家模式，这个概率的分母还要乘 0.85 并向下取整。



生成史莱姆的区域是宽 3840 像素、高 900 像素的矩形；其横向范围是以玩家中心为参照，左右各 1920 像素（不包括右端点）；纵向范围是以玩家中心为参照，上方 900 像素到 1800 像素（不包括上端点）。在这个区域内随机取一个像素，以该像素所在格作为史莱姆生成格。发生下列情形之一的，生成失败。

- 生成格横坐标距离地图左右边界小于 10 格
- 生成格纵坐标在 `0.3worldSurface` 之上
- 生成格纵坐标在 `worldSurface` 之下
- 生成格有人工背景墙
- 以生成格为参照，左右各 3 格，上 5 格，下 2 格的 7×8 矩形区域内有未虚化的实体块（不包括平台类图格）

粉史莱姆的概率为 1/200。在没有生成粉史莱姆的前提下，紫、蓝、绿史莱姆的概率分别为：专家模式 1/7、4/7、2/7，普通模式 1/10、27/50、18/50。

## B.7 特殊判定

### B.7.1 沙块的局部聚簇检查

沙块的局部聚簇检查用于检查某一个图格及其下方是否有足够大的一片沙块区域。如果有，检查通过。

### B.7.2 Boss 与事件检查

Boss 与事件检查用于检查是否有正在进行的 Boss 与事件。如果没有，那么检查通过。Boss 也包括**实际上的 Boss**。该内容对应源码中的 `NPC.AnyDanger()`。



# 附录 NPC 无敌帧机制

---

作者：dhuapskciao

## C.1 引入

众所周知，泰拉瑞亚中存在无敌帧这么一个机制：当你用某些武器攻击到怪物之后，怪物会获得一段时间的无敌帧，在这一段时间内怪物“不会受到伤害”。

因此，用迷你鲨配合陨石子弹时射击某一个怪物时，连续的两发子弹只有一发能击中怪物；因此，用虎爪站撸世纪花时要特别注意不能挂雨云或者召唤小蜘蛛，否则虎爪会打不出应有的伤害。

我们知道那些能穿透敌人的子弹、魔法或剑气一般会给敌人造成无敌帧，许多召唤物也会给怪物造成无敌帧。但是在游戏过程中，细心的玩家可能会发现以上的经验会有些例外，比如说：

- 星尘龙的攻击尽管会造成无敌帧，但不会影响叶绿弹的输出。
- 虽然夜明弹会穿透，但它们之间并不会互相影响输出。这个特性与陨石子弹非常不同。
- 造成无敌帧的子弹、魔法、剑气与召唤物，它们造成无敌帧的时间并非完全一样：比如说棱镜造成的无敌帧时间明显就比鲨龙卷召唤物的碰撞造成的无敌帧时间短很多。

以及还有一些问题：

- 燃烧、霜火、诅咒焰等 debuff 会不会造成骗伤？
- 多人联机时玩家 A 的攻击造成的无敌帧会不会影响玩家 B 的输出？

这一切都需要我们对无敌帧机制进行更深入的研究。

## C.2 说明

### C.2.1 NPC 与射弹

NPC 不仅包含电工妹和哥布林工匠等城镇 NPC，还包含所有的怪物，甚至一部分怪物的攻击方式——比如地牢的黑暗法师打出的水弹，猪鲨吐出的泡泡。本文中以 NPC 代指敌对怪物。

### C.2.2 伤害途径

玩家对 NPC 造成伤害的主要途径有两类：

- 射弹伤害 NPC。



- 用剑等近战武器挥砍。

为了以下行文方便，我们把第一类统称射弹，第二类统称近战武器。

### C.2.3 普通无敌帧的实现机制

游戏中每个 NPC 都拥有一个计时器 `Immune`，这是一个长度为 256 的自然数列。每帧，数列中的每个值都会下降 1，直到 0 为止。玩家  $k$  的许多（具体见下文）攻击方式击中 NPC 时，会检测当前的 NPC 的 `Immune` 的第  $k$  个位置是否为 0，是 0 才能伤害到敌人。于是 NPC 的 `Immune` 中的第  $k$  个数  $n$  表示该 NPC 对玩家  $k$  的无敌帧还会持续  $n$  帧。

所以游戏中不同玩家对 NPC（即使是同一个 NPC）造成的普通无敌帧是独立计算，互不影响的。这是游戏中最普遍的无敌帧。

当然，游戏里还有两类无敌帧，这两类无敌帧与下面会说到的两类特殊射弹绑定，所以详细的之后再谈。

## C.3 射弹造成的无敌帧

与无敌帧有关的射弹属性主要是射弹的穿透能力，在源码里相应的变量是 `Penetrate`。游戏中的每个射弹都拥有自己的 `Penetrate` 值，`Penetrate` 若为正数  $k$ ，则表明它能击中  $k$  个 NPC（穿透  $k - 1$  个），如果是  $-1$ ，则表明它能击中在自己行进路线上的所有 NPC。

`Penetrate` 为正数时，每击中一次 NPC 都会下降 1，到 0 的时候这个射弹就会被销毁。

从这一方面能把射弹分成两类——穿透的（初始的 `Penetrate` 不为 1）和不穿透的（初始的 `Penetrate`=1）。注意，虽然穿透的射弹在还剩一次击中 NPC 的能力时 `penetrate`=1，但还是被认为是穿透型射弹。

射弹还有两个标签：

- `usesLocalNPCImmunity`
- `usesIDStaticNPCImmunity`

游戏中的 700 多种射弹可以根据这俩标签的有无被分成三类：

- 第一类是这俩标签都没有（被设定为 `False`）的。
- 第二类是 `usesLocalNPCImmunity=True` 的。
- 第三类是 `usesIDStaticNPCImmunity=True` 的。

第一类射弹占了绝大多数，第二类仅仅有一小部分，第三类仅有一种：即 699 号射弹，食人魔掉落的关刀的本体。

这三类射弹计算无敌帧、以及被无敌帧影响时的机制不尽相同。

### C.3.1 第一类射弹

第一类射弹内的大多数穿透射弹在穿透 NPC（击中当前 NPC 的时候 `Penetrate` 不是 1）时，会对 NPC 造成 10 帧的无敌帧。但有一些例外，见下表：

表 C.1: 特殊的第一类射弹

射弹 ID	射弹	Immune(帧)
632	棱镜激光	5
514	钉枪的钉子	3
595	Arkhalis	5
625-628	星尘龙	6
286	爆炸子弹	5
443	电子球发射器的射弹	8
424-426	陨石魔杖打出的陨石	5
634-635	星云烈焰	5
659	神灯烈焰	5
246	毒刺发射器射弹	7
249	毒刺发射器射弹的碎片	7
190	机械食人鱼	8
409	利刃台风	6
407	鲨龙卷召唤物本体	20
311	玉米糖	7
582	机械师扳手	7

注意鲨龙卷召唤物指的是鲨龙卷本体的碰撞。至于它发射出的小鲨鱼，那东西不穿透也不造成无敌帧。

当第一类射弹碰到 NPC 时，如果 NPC 对当前玩家的 Immune 为 0，或者射弹是不穿透型射弹，那么就能击中 NPC，否则不行。第一类射弹的特点是互相影响，比如陨石子弹 A 对 NPC 造成的无敌帧会影响陨石子弹 B 的杀伤。当然，非穿透射弹不受影响，所以如果你用泡泡枪站撸世花，就不需要担心世花骗伤的问题；用枪械配合叶绿弹打月总时，召唤一条星尘龙出来也不会影响输出。

### C.3.2 第二类射弹

表 C.2: 第二类射弹

射弹 ID	射弹	LocalNPCImmunity	Immune
611	日耀喷发剑本体	6	4
612	日耀喷发剑爆炸	6	4
617	星云奥秘本体	8	0
618	星云奥秘爆炸	8	0
638	月明弹	-1	0
639	月明箭	-1	0
640	月明箭尾迹	-1	0

射弹 ID	射弹	LocalNPCImmunity	Immune
642	月亮传送门激光	10	0
645	月耀	-1	0
656	禁戒套的龙卷风	8	0
661	黑玛瑙爆破枪的黑玛瑙	8	0
664,666,668	烈焰炮台射弹	-1	0
680	弩车炮台射弹	-1	0
688-690	闪电光环	3	0
694-696	陷阱炮台的爆炸	30	0
697	瞌睡章鱼本体	12	0
698	瞌睡章鱼爆炸	-1	0
700	关刀打出的龙头	-1	0
704	无限智慧巨著的龙卷风	-1	0
706	幽灵凤凰	10	0
707	天龙之怒旋转	6	0
708	天龙之怒射弹	6	0
710	空中克星箭	-1	0
711	贝特西之怒	-1	0

第二类射弹被发射出去后，会拥有一个计时器 `localNPCImmunity`，这是一个长度为 200 的整数列，其中的第  $k$  个值表明这个射弹对序号为  $k$  的 NPC 的“局部无敌时间”。这个整数列中的所有大于 0 的值每帧都会减少 1，直到 0 为止。而小于 0 的值则不变。当第二类射弹碰到 NPC 时，游戏会首先检测这个射弹对这只 NPC 的局部无敌时间是否为 0。如果不是，则无法击中；如果是，则继续检测 NPC 对发射这枚射弹的玩家的 `Immune`（即普通的无敌帧）是不是 0，以及这个射弹是不是非穿透射弹，如果以上至少有一个是成立的，则最终击中 NPC。第二类射弹击中 NPC 时，绝大多数不会给 NPC 造成普通无敌帧，唯一的例外是喷发剑，上表里已经显示出来了。

注意，局部无敌时间是射弹的属性（而普通的无敌帧是 NPC 的属性），游戏中每一个射弹都拥有独立计算的局部无敌时间，加之绝大多数第二类射弹不会造成普通的无敌帧，这意味着第二类射弹基本上不会互相影响（唯一的例外是喷发剑）。所以说虽然月明弹是穿透的，但它们并不会像陨石弹那样连续发射的两发子弹只能击中一发。

此外，还可以注意到一部分第二类射弹的 `localNPCImmunity` 是 -1，这意味着这个射弹无法再次击中已经击中过的 NPC，而那些 `localNPCImmunity` 大于 0 的抛射体，则表明它在击中某个 NPC 之后一段时间，可以再次击中这只 NPC（典型的就是一些星云奥秘）。

以及由于第二类射弹碰到 NPC 时仍然会检测 NPC 的 `Immune`，所以普通无敌帧还是

会影响第二类射弹的输出。简单来说，虽然第二类射弹基本不会互相影响，但它们会受到第一类射弹的影响。

### C.3.3 第三类射弹

只有那个 ID 为 699 的大关刀的本体。

与那个东西有关的是 `perIDStaticNPCImmunity`，这个变量是个 714（游戏内射弹总种类数） $\times$ 200 的整数矩阵，这是个游戏的内置变量，不属于任何一个 NPC 或者射弹。关刀碰到序号为  $k$  的 NPC 时，会检测矩阵第 700 行，第  $k + 1$  列的数字是否小于等于 `GameUpdateCount`，如果是则继续检测 NPC 对发射这枚射弹的玩家的 `Immune` 是不是 0，以及这个射弹是不是非穿透抛射体，如果以上至少有一个是成立的，则关刀最终击中 NPC。

关刀击中 NPC 后，也会在矩阵的相应位置赋值 `GameUpdateCount+36`。这个大矩阵内的数值本身是不会下跌的（不像 `Immune` 和 `localNPCImmunity`），但因为 `GameUpdateCount` 会上涨（速率应该也是每帧 1），所以起到的倒计时效果一样。

所以关刀的特性和第二类射弹差不多，但有一点不一样：不同玩家的关刀会互相影响输出。

## C.4 近战武器造成的无敌帧

游戏每帧都会检测每个 NPC 对某个玩家的 `Immune` 是不是 0，以及玩家的 `attackCD` 是不是 0，如果以上都成立，并且武器碰到了 NPC，则对 NPC 造成伤害。玩家的 `attackCD`= 玩家 `itemAnimationMax/3`。而 `itemAnimationMax` 是对应武器的 `useAnimation*` 玩家的 `meleespeed`（为实际近战攻速的倒数），`useAnimation` 是武器的自身属性。所以实战中，挥舞一次剑能打到的 NPC 数量是有限的。并且挥舞武器受普通无敌帧的影响。

此外，当挥舞武器给 NPC 造成伤害时，会给 NPC 对当前玩家的一个比较短的普通无敌帧，持续时间是当前玩家的 `itemAnimation`。这个数值在你刚开始挥舞武器的时候与 `itemAnimationMax` 一样，然后下降，如果没理解错的话，当你挥舞完成的时候，这个值会降到 0。这个无敌帧的存在应当只是为了保证玩家每一次挥砍，对同一个 NPC 只能造成一次伤害。

## C.5 零散内容

以下是一些零散内容：

- 着火、诅咒焰以至于星尘细胞吸附、破晓等烧血 debuff 通过 `NPC.UpdateNPC_Buff ApplyDOTs` 函数直接作用于 NPC 生命值，不受无敌帧影响也不造成无敌帧。
- 部分射弹在击中 NPC 后属性会改变——主要是那些击中目标后爆炸的射弹比如爆炸子弹、月耀等。改变的属性主要是射弹的大小，不过有时候 `Penetrate` 属性也会变，这造成了一些有趣的现象：爆炸子弹自身 `Penetrate` 是 1，但击中 NPC 后 `Penetrate`

变成-1，长宽变成 80 像素然后再造成一下伤害。于是爆炸子弹本体伤害不受无敌帧影响，但是爆炸的范围伤害会受到无敌帧影响。同样的现象在星云烈焰上也能观察到（星云烈焰的爆炸是范围伤害，范围是 50\*50 像素）。

- 岩浆对 NPC 造成 30 帧的普通无敌帧，用 NPC 的 Immune 列表的第 256 个数储存并计算，游戏中能存在的玩家总数为 255 因此不会影响玩家的输出。
- 陷阱也会对 NPC 造成无敌帧，并且会被算到玩家头上，因此会影响玩家的输出。

## 附录 液体

泰拉瑞亚中的每个图格都包含了液体属性，液体属性包含液体量和液体种类。液体量是 0 到 255 的一个整数，0 表示没有液体，255 表示满液体。

泰拉瑞亚中有两个液体数组，一个是活跃液体，大小为 5000；一个是缓冲区，大小 10000。每当一格液体有潜在的流动可能性，这一格就会被加入液体数组。每帧中液体数组会进行刷新，如果某一格不再有流动的可能性，这一格就会被从液体数组中清除。与射弹数组不同，液体数组始终是从头连续填充的。当液体数组中一格被清除时，空位由液体数组中最后一格液体取代，从而保持连续填充的特性。

活跃液体数量上限与活跃液体数组长度不同，前者是软上限，后者是硬上限。活跃液体数量上限等于  $2500 + 2500 \times \text{gfxQuality}$ 。

### D.1 全图液体刷新

全图液体刷新有三种模式：恐慌模式、快速模式和正常模式。

#### D.1.1 恐慌模式

如果活跃液体和缓冲区中总格数连续 1800 次刷新（30 秒）都超过了 4000 格，或者单次刷新中活跃液体和缓冲区总格数超过了 13500 格，那么进入恐慌模式。恐慌模式会覆盖快速安置模式和正常模式。

在恐慌模式中，从  $\text{maxTilesY} - 3$  开始，从下到上**快速安置**5 行<sup>1</sup>，并把活跃液体和缓冲区全部清空。

**液体快速安置** 该过程有三个参数：`verbose`<sup>2</sup>，`minY` 和 `maxY` 分别表示刷新区域的上下界。这三个参数都是可选参数，`verbose` 缺省值为 0，`maxY` 的缺省值为  $\text{maxTilesY} - 3$ ，`minY` 的缺省值为 3。返回值是安置过程中发生变化的液体格数（不包括仅进行了横向安置与纵向安置其一的情况；多次刷新时多次计数）。

快速安置时从下到上逐行刷新。每行刷新三次，第一次和第三次从左到右刷新，第二次从右到左刷新。刷新的两端横坐标分别为 2 和  $\text{maxTilesX} - 2$ 。每次刷新时刷新起点不刷新终点<sup>3</sup>。

进行单格液体安置时，纵向安置与横向安置交替运行。纵向安置时，将液体下移直至遇到障碍。横向安置移动方向优先选择当前的刷新方向，横移直至遇到障碍；如果向当前刷新方向一格都移动不了，则尝试向反向移动。

<sup>1</sup>实际上当  $\text{maxTilesY} < 10$  时会有出入，但是这种极端情况我们不考虑。

<sup>2</sup>暂时不知道 `verbose` 表示什么，但是可以确定的是它不影响本书中讨论到的内容。

<sup>3</sup>也就是说每行刷新结束时，左端点刷新了 2 次，右端点刷新了 1 次，其他点都刷新了 3 次。



首先进行纵向安置尝试, 尝试结束后, 如果本格仍能向下格流动<sup>4</sup>, 那么向下格填充。然后进行横向安置, 横向安置时每移动一格后都会插入一个纵向安置尝试与向下格填充尝试, 如果成功下移, 那么重新进行横向安置。如果某次下格填充尝试后本格液体耗尽, 那么安置完成。否则只有当纵向安置和横向安置全部失败时, 安置才会完成。

安置完成后, 如果上下左右 (优先级左 > 右 > 上 > 下, 只对其中一格执行) 其一与本格可反应且反应物之一是熔岩, 那么对熔岩格执行**熔岩反应判定**。

如果本格仍有液体, 与上一段类似, 执行**蜂蜜反应判定**。

### D.1.2 快速模式与正常模式

快速模式的触发条件是活跃液体多于 2000 格。快速模式与正常模式都是对活跃液体数组中的一部分从前到后依次进行**刷新**, 然后进行后续判定。它们的区别是, 快速模式中每个刷新对象的 `delay` 值都会被赋值为 10, 这样它们都会快速流动, 无论液体黏性如何; 而正常模式中黏性会影响流速<sup>5</sup>。另一个区别是, 快速模式中不会跳过刷新, 而正常模式中可能有一些液体会跳过刷新, 关于跳过刷新的机制, 请参考**单格活跃液体刷新**。

快速模式或正常模式每执行一次, 会在 `wetCounter` 中计数。

我们已经提到过, 快速模式与正常模式都只是刷新一部分液体。这一部分液体是怎么计算出来的:

- $\text{num1} = \text{活跃液体数量上限} / \text{刷新周期}$ , 这个除法是整型除法。
- $\text{num2} = \text{num1} \times (\text{wetCounter} - 1)$ 。
- $\text{num3} = \text{num1} \times \text{wetCounter}$ 。
- 如果  $\text{num3} > \text{活跃液体数量上限}$ , 那么  $\text{wetCounter} = \text{刷新周期}$ 。
- 如果  $\text{wetCounter} = \text{刷新周期}$ , 那么  $\text{num3} = \text{活跃液体数量上限}$ 。
- 刷新活跃液体数组中从  $\text{num2}$  到  $\text{num3}-1$  这一段。

在执行完刷新后, 如果 `wetCounter` 达到了刷新周期<sup>6</sup>, 就会在刷新后进行后续判定。刷新周期等于  $17 - 10\text{gfxQuality}$ , 向下取整。可能的取值有 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17。

**后续判定** 将 `wetCounter` 归零。从后往前扫描活跃液体数组, 将 `kill` 值大于 4 的全部**删除**。将缓冲区的前若干格从前往后依次加入活跃液体数组并从缓冲区中删除, 这个数量等于执行该步之前的活跃液体与缓冲区液体数量最大值<sup>7</sup>。

然后进行阻塞判定。简单地讲, 当活跃液体维持在某个数量附近很长一段时间, 那么就进入阻塞模式, 将活跃液体强行删除。阻塞判定由阻塞时间与阻塞格数控制。如果有至少一格活跃液体, 并且活跃液体格数与阻塞格数相差小于 50 格, 那么阻塞时间加一; 如果阻塞时间达到了 10000, 那么在阻塞模式中从后往前**删除**所有活跃液体然后将阻塞

<sup>4</sup>下格不满且与本格液体相同。

<sup>5</sup>详情见**单格活跃液体刷新**。

<sup>6</sup>注意 `wetCounter` 不仅会每次递增, 还有可能在  $\text{num3} > \text{活跃液体数量上限}$  时被直接赋值。

<sup>7</sup>源码如此。



时间归零。在其他情况下，即活跃液体数组为空或者活跃液体格数与阻塞格数相差至少 50 格，那么将阻塞格数赋值为活跃液体格数并将阻塞时间归零。

## D.2 将某格加入活跃液体数组

如果这一格已经在活跃液体数组中，或者距离地图边缘小于 5 格，或者没有液体，那么不加入。

如果达到了活跃液体数量上限，将该格加入液体缓冲区。

如果未达到活跃液体数量上限，将该格加入活跃液体数组中第一个空位<sup>8</sup>。如果该格中的图格会被对应的液体破坏，那么破坏。

## D.3 单格活跃液体刷新

如果本格是可以阻挡液体的图格，kill 值设为 9，刷新结束。

如果本格是水并且距离世界底端小于 200 格，液体量减 2。

如果本格液体量为 0，kill 值设为 9，刷新结束。

如果本格是熔岩，执行熔岩反应判定。如果不是快速安置液体状态，delay 值小于 5 则加 1，达到 5 则清零，前一种情况下刷新直接结束。也就是说这一步每执行 5 次才会进行一次刷新。

如果本格不是熔岩，将左右上下四格中是熔岩的图格按顺序加入活跃液体数组。如果本格是蜂蜜，执行蜂蜜反应判定；如果不是快速安置液体状态，delay 值小于 10 则加 1，达到 5 则清零，前一种情况下刷新直接结束。也就是说这一步每执行 10 次才会进行一次刷新。如果本格是水，将左右上下四格中是蜂蜜的图格按顺序加入活跃液体数组。

处理纵向流动。如果下格不阻挡液体，与本格液体相同且不满，那么向下流动，把下格加入活跃液体数组，并且本格和下格会跳过下次**正常刷新**。流动后，如果本格液体量大于 250，则改为 255，否则将左右两格按顺序加入活跃液体数组。

如果本格还有液体，处理横向流动。对左右若干图格，判断是否满足不阻挡液体且和本格液体相同的条件，分为四种情况。

- 左右各三格都满足条件。将本格和左右各三格的液体量设为这七格液体量的平均值，四舍五入<sup>9</sup>。将左右各三格中液体量变化的图格加入活跃液体数组，顺序是左一、右一、左二、右二、左三、右三。如果本格液体量变化了，将左右各三格全部加入活跃液体数组，顺序同上。如果左右各三格液体量都没有变化并且上格有液体，本格液体量还原到变化之前的值。
- 左右各两格都满足条件，但是左三和右三至少有一格不满足条件。对本格和左右各两格进行处理，处理方法和第一条类似。
- 左右各两格中左一和右一满足条件，左二或右二不满足条件。将本格和左右各两格中满足条件的图格的液体量设为这几格液体量的平均值，四舍五入。将左右各两格

<sup>8</sup>因为液体数组是连续填充的，第一个空位就在最后一个活跃液体之后。

<sup>9</sup>源码中实际的处理方式是四舍六入五留双，但是对于七格平均的情况，该舍入方式等同于四舍五入。

中液体量变化的图格加入活跃液体数组，顺序是左一、右一、左二、右二。

- 左一和右一有且仅有一个满足条件。将本格和满足条件的一格的液体量设为这两格液体量的平均值，四舍五入。将左右两格中液体量变化的图格加入活跃液体数组。

如果在以上全过程中，本格的液体量从 255 变成了 254，恢复到 255。

如果在以上全过程中，本格液体量不变，kill 值增加 1，否则 kill 值归零。

## D.4 熔岩反应判定

对于某格熔岩，如果上下左右四格均无可反应的液体，判定结束。如果上左右三格中至少有一格有可以反应的液体，那么进行第一部分判定；否则进行第二部分判定。

**第一部分判定** 这部分判定解决熔岩与上左右三格反应的情况。首先吸收上左右三格中全部的非熔岩液体。如果吸收液体总量小于 24，那么不反应，判定结束。否则该格有三种情况：

- 如果该格有可被黑曜石破坏的图格，那么被破坏。
- 如果该格有不可被黑曜石破坏的图格，那么判定结束。
- 如果该格为空，不执行操作。

然后把该格中的液体清空，生成反应块。只要吸收前，上左右三格中有至少一格蜂蜜，就生成松脆蜂蜜块，否则生成黑曜石。

**第二部分判定** 这部分判定解决熔岩与下格反应的情况。首先分为四种情况：

- 下格是可以被黑曜石或武器破坏的图格，被破坏。
- 下格不是宝箱或梳妆台，本格是宝箱或梳妆台，不执行操作。
- 下格不是可以被黑曜石或武器破坏的图格，并且不是宝箱梳妆台那种情况，判定结束。
- 下格没有图格，不执行操作。

如果本格液体量小于 24，直接删除，判定结束。否则进行反应，删除本格和下格的液体，在下格生成反应块。

**蜂蜜反应判定** 蜂蜜反应判定与熔岩反应判定类似。不同的地方是：

- 吸收液体量的阈值为 32 而不是 24。
- 第一部分判定中，上左右三格有至少一格岩浆，就生成松脆蜂蜜块，否则生成蜂蜜块。
- 第二部分判定中，没有宝箱梳妆台那种判定。只要下格不是可以被黑曜石或武器破坏的图格，判定结束。

## D.5 将某格从活跃液体数组中删除

当游戏推断活跃液体数组中的某格不再流动时，就会把这一格从活跃液体数组中删除。在删除前需要进行一系列结算。

如果本格液体量小于 2，那么清零，并且在这种情况下，如果左格或右格液体量小于 2，也清零。左格和右格归零时还会被加入活跃液体数组。

如果本格液体量为 2 到 19，并且可以往左右下三格之一流动<sup>10</sup>，那么本格液体清零。

如果本格液体量至少为 20，并且可以往下格流动，那么本格的 kill 值为 0，删除失败，结算结束。在**阻塞模式**中会跳过这一条。

如果本格液体量为 20 到 249，并且上格有液体，那么将上格加入活跃液体数组。

在本格液体还没有被清零的情况下，如果左格有液体并且左下格液体量小于 250，将左格加入活跃液体数组；如果右格有液体并且右下格液体量小于 250，将右格加入活跃液体数组。如果本格液体是熔岩，执行熔岩反应判定，然后烧掉周围八格的草皮。如果本格液体是蜂蜜，执行蜂蜜反应判定。

以上过程完成以后，就可以把活跃液体数组这个位置删除了。删除的时候，为了保持液体数组连续填充的特性，这一格的位置会由最后一格活跃液体代替。然后如果本格有草药，进行草药破碎检查。

**草药破碎检查** 如果下格不是供草药正常生长的图格，草药破碎。除火焰花之外，如果本格有熔岩，草药破碎。对于火焰花，如果本格有水或蜂蜜，草药破碎。如果火焰花本格有熔岩，那么液体量小于 16 时，火焰花不开花；否则成熟的火焰花会开花。

需要注意的是，这部分检查并不意味着在游戏中成熟的火焰花不会被熔岩破坏。具体原因还有待研究。

---

<sup>10</sup>左/右格流是指左/右格液体量小于本格，并且左格允许液体通过。可以往下格流是指下格液体不满，并且允许液体通过。

# 附录 泰拉瑞亚中的常用变量与特殊集合

## E.1 常用变量

变量名	变量值	意义
sWidth	1920 像素	标准屏幕宽度
sHeight	1080 像素	标准屏幕高度
activeRangeX	4032 像素	活跃区域宽度
activeRangeY	2268 像素	活跃区域高度
LogicCheckScreenWidth	1920 像素	逻辑屏幕宽度
LogicCheckScreenHeight	1200 像素	逻辑屏幕高度

变量名	spawnRangeX	spawnRangeY	safeRangeX	safeRangeY
意义	生成区域宽度	生成区域高度	安全区域宽度	安全区域高度
玩家装备着步枪瞄准镜或狙击镜，没有拿狙击步枪	114	77	92	65
玩家拿着狙击步枪，没有装备步枪瞄准镜或狙击镜	124	87	102	75
玩家既装备着步枪瞄准镜或狙击镜又拿着狙击步枪	132	95	110	83
其他	84	47	62	35

表 E.2: 生成区域与安全区域

## E.2 图格集合

### E.2.1 环境方块

环境	图格名称	图格贴图
神圣	神圣草皮、矮小神圣植物、高大神圣植物、珍珠沙块、珍珠石块、粉冰雪块、硬化珍珠沙块、珍珠沙岩块	

环境	图格名称	图格贴图
腐化	腐化草皮、腐化植物、黑檀石块、腐化多刺灌木、黑檀沙块、紫冰雪块、硬化黑檀沙块、黑檀沙岩块	
猩红	猩红草皮、红冰雪块、猩红石块、猩红沙块、猩红多刺灌木、硬化猩红沙块、猩红沙岩块	
雪原	雪块、雪砖、冰雪块、薄冰、紫冰雪块、粉冰雪块、红冰雪块	
丛林	丛林草皮、矮小丛林植物、丛林藤蔓、高大丛林植物、丛林蜥蜴砖	
蘑菇	蘑菇草、发光蘑菇、地下巨型发光蘑菇	
陨石	陨石	
地牢	蓝砖、绿砖、粉砖	
沙漠	沙块、黑檀沙块、珍珠沙块、猩红沙块、硬化沙块、硬化黑檀沙块、硬化珍珠沙块、硬化猩红沙块、沙岩块、黑檀沙岩块、珍珠沙岩块、猩红沙岩块	

E.2.2 实体块

E.3 背景墙集合

E.3.1 透光墙

天空、玻璃墙、木栅栏、铅栅栏、乌木栅栏、红木栅栏、珍珠木栅栏、暗影木栅栏、铁栅栏、针叶木栅栏、棕榈木栅栏、彩纸墙

E.3.2 天然背景墙

天然土墙、黑檀石墙、天然蓝砖墙、天然绿砖墙、天然粉砖墙、天然狱石砖墙、天然黑曜石砖墙、天然泥墙、珍珠石墙、雪墙、紫晶墙、黄玉墙、蓝玉墙、翡翠墙、红玉墙、钻石墙、独特洞壁 1/2/3/4/5/6/7/8、蜘蛛墙、天然草墙、天然丛林墙、天然花墙、腐化草墙、神圣草墙、冰墙、黑曜石背景墙、天然蘑菇墙、猩红草墙、猩红石墙、天然蜂巢墙、天然丛林蜥蜴砖墙、天然蓝板墙、天然蓝瓷砖墙、天然粉板墙、天然粉瓷砖墙、天然绿板墙、天然绿瓷砖墙、铁栅栏、船帆、洞壁 1/2、天然大理石墙、天然花岗岩墙、沙岩墙、腐化墙 1/2/3/4、猩红墙 1/2/3/4、土墙 1/2/3/4、神圣墙 1/2/3/4、丛林墙 1/2/3/4、熔



岩墙 1/2/3/4、岩石墙 1/2/3/4、硬化沙墙、硬化黑檀沙墙、硬化猩红沙墙、硬化珍珠沙墙、黑檀沙岩墙、猩红沙岩墙、珍珠沙岩墙

和天然背景墙相对的是人工背景墙。

## E.4 NPC 集合

### E.4.1 Boss

### E.4.2 实际上是 Boss

星旋柱、星尘柱、星云柱、日曜柱、火星探测器

## 附录 常用工具与网址



### F.1 Wiki

[英文](#) | [中文](#) | [中文镜像站](#) | [英文 apk](#)

### F.2 论坛

[官方论坛-\[T-MEC\]](#) [电路工程师联盟](#) | [T-MEC 帖子存档](#) | [中文论坛](#) | [CurseForge 地图下载](#)

### F.3 地图编辑器

[主页](#) | [下载 1](#) | [下载 2](#) | [推荐教程\(1\)\(2\)\(3\)\(4\)](#)

### F.4 模组

#### F.4.1 tModLoader

[官网](#) | [GitHub](#) | [下载](#) | [Wiki](#) | [文档](#)

#### F.4.2 CheatSheet

[主页](#) | [下载 1](#) | [下载 2](#) | [发布页](#)

#### F.4.3 HERO's Mod

[主页](#) | [下载 1](#) | [下载 2](#) | [发布页](#)

#### F.4.4 MechScope

[主页](#) | [下载](#) | [发布页](#) | [中文介绍](#)

### F.5 源码

[1.3.5 各版本](#) | [1.3.2.1](#) | [1.3](#) | [1.2.4.1](#) | [1.2.0.3.1](#) | [Xbox 360](#)



## 附录 你应该知道的黑科技

---

### G.1 TNT 神教

使用地图编辑器或 mod 将炸药或地雷放在宝箱下可以无限引爆。

### G.2 传送器浮空

将家具放在传送机上，然后挖掉传送机下方的支撑方块，家具就可以浮空。有一些例外。

### G.3 传送枪加速

手持传送枪时水平速度不变且无视上限，竖直速度有加成。经过传送门时速度大小不变。所以可以通过传送门将所有速度转化为水平速度，再利用重力提高竖直加速度从而增加总速度，反复进行这个过程，理论上可以无限加速。但是由于物理帧判定问题，实际操作非常困难，用简单装置只能加速到 500mph 左右。

## 附录 高质量电路作品视频

---

### H.1 视觉效果

- Terraria Bad Apple!! <https://www.bilibili.com/video/av46694445>
- 【泰拉瑞亚】彩虹小电视我们起飞 <https://www.bilibili.com/video/av16601896>
- [Terraria] 官方推荐电路作品-Bad apple(音乐为后期添加) <https://www.bilibili.com/video/av22343683>
- 【Terraria 电路】像素盒定格动画兔子 <https://www.bilibili.com/video/av50343156>
- [Terraria] Digi-Comp 2 装置展示-Zerogravitas <https://www.bilibili.com/video/av22690346>
- 泰拉瑞亚有趣的矿车 <https://www.bilibili.com/video/av5271577>
- 【Terraria 电路】给女朋友的生日礼物 <https://www.bilibili.com/video/av21009075/>
- Terraria in-game Pixel Art Animation <https://www.bilibili.com/video/av5301176/?p=5>
- Terraria 莫尔条纹动画兔子 <https://www.bilibili.com/video/av49966963>
- Terraria 电子钟 <https://www.bilibili.com/video/av55613747>
- Terraria 异世界四重奏 ED『異世界ガールズ トーク』（传送器定格动画） <https://www.bilibili.com/video/av54977071>

### H.2 解谜冒险

- [Terraria] 障碍挑战地图-Mappygaming <https://www.bilibili.com/video/av22787315>
- 泰拉瑞亚 terraria 超难创意解谜地图通关介绍（上） <https://www.bilibili.com/video/av19793617>
- 泰拉瑞亚 terraria 超难创意解谜地图通关介绍（中） <https://www.bilibili.com/video/av20101902>
- 泰拉瑞亚 terraria 超难创意解谜地图通关介绍（下） <https://www.bilibili.com/video/av20524074>
- 泰拉瑞亚在游戏里面玩游戏 <https://www.bilibili.com/video/av6594668>
- [Terraria] 使用传送枪达到最大速度!3000+mph! <https://www.bilibili.com/video/av24580434>

- 【Terraria】【FuryForged】冒险地图 Harbinger 通关实况 <https://www.bilibili.com/video/av37553671/>
- 在 Terraria 中玩俄罗斯方块!? <https://www.bilibili.com/video/av38924330>
- 在 Terraria 中解魔方? (Read Description) <https://www.bilibili.com/video/av56760618>

### H.3 刷怪刷物品

- [Terraria] 半砖松露蠕虫农场 (390+ 条/小时) !-DicemanX <https://www.bilibili.com/video/av23029412>
- 泰拉瑞亚-诈个尸 <https://www.bilibili.com/video/av13411383>
- 泰拉瑞亚雕像回血裸装专家月总 <https://www.bilibili.com/video/av6393957>
- 【Terraria 电路】【Joe Price】全自动刷怪场、刷 BOSS 合集专家模式 <https://www.bilibili.com/video/av32865707/>
- Terraria 1.3.1 Pumpkin Moon Final Wave at 806 pm, 255 platinum coins in a night <https://www.bilibili.com/video/av5356226/?p=5>
- Terraria 1.3 Frost Moon Final Wave at 856 pm, 75820 Total Points - World Record <https://www.bilibili.com/video/av5356226/?p=6>
- 【泰拉瑞亚】最快的 boss 击杀集锦无法被超越的世界纪录? <https://www.bilibili.com/video/av20725652>
- 南瓜神教-旧日军团 <https://www.bilibili.com/video/av10885401/?p=3>
- 南瓜神教-霜月夜 <https://www.bilibili.com/video/av10885401/?p=4>
- 【Terraria 电路】改进的高效全自动树场 <https://www.bilibili.com/video/av38882621/>
- Terraria 刷石机 <https://www.bilibili.com/video/av50203346>
- Terraria 水晶碎块农场 <https://www.bilibili.com/video/av57287619>

## 附录 待实现的电路或装置

---

### I.1 电子钟

原理很简单，使用假人驱动计数器。需要实现功能：精确到游戏秒，显示月相，进地图启动，使用日晷后会校正时间。可考虑的功能：12 小时制与 24 小时制转化，自定义闹钟。

尽管有人已经做了一个电子钟<sup>1</sup>，但是它没有实现 12 小时与 24 小时转化功能与闹钟功能，也没有精确到游戏秒。

### I.2 确定有限状态自动机

虽然确定有限自动机的组合逻辑构造与状态转换表有关，但是自动机的整体结构还没有一个优秀的可用方案。主要的难点可能在于接线，因为自动机中的线路是循环的，这违背了泰拉瑞亚中逻辑电路的一般规律。

### I.3 俄罗斯方块

像素盒显示器的宽度限制是非常讨厌的，这使它无法实现我们一般习惯的横向显示器。另一方面，由于宽度被限制在 24 格之内，其实际能显示的东西非常有限。俄罗斯方块是为数不多的像素盒显示器可以胜任的任务。经典的俄罗斯方块主显示屏宽 10 格，高 20 格。副显示屏用来显示下一个方块，2\*4 或 4\*2 皆可。使用方向感应器控制。

有人已经做过俄罗斯方块了<sup>2</sup>，但是它不能变速，不能计分，操作也不方便。

### I.4 贪吃蛇

贪吃蛇是像素盒显示器能胜任的另一个任务。其缺点是像素盒显示器是单色的，可能无法分辨身体和果实。屠宰场与 TheRedStoneCrafter 已经分别做了一些工作。

### I.5 魔方

使用彩线灯泡做显示。

有人已经做过魔方了<sup>3</sup>，但是它操作极不方便。

---

<sup>1</sup><https://www.bilibili.com/video/av55613747>

<sup>2</sup><https://www.bilibili.com/video/av38924330>

<sup>3</sup><https://www.bilibili.com/video/av56760618>

## I.6 Flappy Bird

虽然 TheRedStoneCrafter 已经做过了 Flappy Bird<sup>4</sup>，但是其效果并不好，还原度可以更高。首先是人物控制，可以利用水 + 海神贝壳模拟 Flappy Bird 中的运动，用尖刺或木尖刺模拟障碍物，检测人物位置，当人物被击退时游戏结束。

虽然使用递次电路可以实现，但是其占地过大，使得纵向上每个单元至少占三格。使用 1 秒计时器串联是最节省空间，静态视觉效果最好的，但是其频率过低。

可以用蜂蜜中的飞镖模拟障碍物，但是飞镖在蜂蜜中几乎不可见，可以使用弹幕踏板激活其他光源或虚实化尖刺来提高视觉效果。使用飞镖的好处就是电路难度较低，只需要设计发射飞镖的电路，其余的动画效果都可以自动完成。

## I.7 2048

使用四向操纵板操作。因为多位数显面积太大，用指数表示，0 表示 2，1 表示 4，2 表示 8，……，9 表示 1024，A 表示 2048。如果要超过 2048，用 b 表示 4096，c 表示 8192，d 表示 16384，E 表示 32768，F 表示 65536，G 表示 131072（可能的最大值）。

使用像素盒显示可以显示出移动动画，但是难度相当大。

## I.8 计算机

---

<sup>4</sup><https://www.bilibili.com/video/av24265449>

# 附录 冷却时间机制

有不少电路物品都有冷却时间。游戏使用一个大小为 1000 的列表（以下称为冷却列表）存储冷却中的图格及其剩余冷却时间，每帧中，整个列表中的剩余冷却时间减 1，如果减到 0 则将该图格从冷却列表中删除。当有冷却时间的图格被激活，游戏尝试将该图格加入冷却列表，如果该图格已经存在冷却列表中，或者冷却列表已满，则加入失败，该图格激活失败。所以，不光图格正在冷却时无法激活，在整个冷却列表已满时也无法激活。

使用冷却时间机制的图格及其冷却时间如下表：

图格名称	冷却时间/帧
炮台	30
雪球发射器	10
烟花盒	30
烟花喷泉	30
矿车轨道	5
飞镖机关	200
超级飞镖机关	200
烈焰机关	200
长矛机关	90
尖球机关	300
喷泉（机关）	200
刷怪雕像	30
刷物品雕像	600
传送 NPC 雕像	300

此外，还有一些图格没有冷却时间，但是也借用了冷却列表用来做倒计时，包括计时器和引爆器。

## 附录 机关射弹的生成和刷新机制

射弹生成中的参数包括射弹坐标、射弹速度、射弹 ID、伤害、击退，和其他所需参数。未经说明的情况下，长度单位为像素，时间单位为帧。例如，速度的单位为像素/帧。

游戏中活跃的射弹被存储在一个大小为 1000 的列表中。新生成射弹时，会将新的射弹放入列表中第一个空位；如果列表已满，则射弹不会生成（但是不代表与射弹生成并列的其他操作也会中止）。

射弹是通过每帧中的刷新操作前进的。所谓刷新操作，就是根据射弹当前的属性计算出射弹下一帧的属性并更新。例如，已知射弹当前坐标和射弹速度，则射弹下一帧的坐标是当前坐标 + 速度。不同的射弹在更新时做出的判断和操作各不相同，甚至某些射弹在一帧中会刷新多次。非穿墙的射弹每次刷新时，如果与某个青绿压力垫板碰撞，并且在刷新前未与该青绿压力垫板碰撞，那么会激活该青绿压力垫板。

射弹的销毁操作会将列表中该射弹位置清空。销毁操作发生的场合主要有：不穿墙的射弹击中实体块；射弹存在时间达到了生存期；射弹离开了地图；有限穿透的射弹达到了穿透上限。

之所以在电路教程里说这么详细，是因为有一些奇怪的情况需要解释。

首先需要注意的一点是，射弹激活青绿压力垫板只发生在刷新后，而不发生在生成后。而刷新时激活青绿压力垫板的条件包括“刷新前未碰撞”。这意味着如果一个射弹在生成时与青绿压力垫板碰撞，那么既不在生成后激活，又不满足刷新时的激活条件，则这个青绿压力垫板不会被激活。

一个更极端的例子是飞镖机关-青绿压力垫板的连锁反应。

### K.1 匀速直线运动的射弹及其速度大小

飞镖机关的飞镖速度 12，超级飞镖机关的飞镖速度 12，彩纸炮的引导速度 14，传送枪台的传送枪子弹速度 93。

### K.2 抛射型射弹的运动机制

电路产生的抛射型射弹有大炮和兔兔炮的炮弹、雪球发射器的雪球。这类抛射型射弹在生成后首先匀速直线运动一段时间（大炮和兔兔炮为 17，雪球发射器为 19），然后获得一个垂直方向的加速度（大炮和兔兔炮为 0.28，雪球发射器为 0.3），并且水平速度依指数衰减（大炮和兔兔炮为 0.99，雪球发射器为 0.98）。

### K.3 炮台发射的射弹的初速度方向

炮台一共有 9 个朝向，其对应的速度方向如??所示。



## K.4 射弹碰撞箱及初始生成位置

## K.5 实验测定抛射射弹轨迹

## 附录 网站代号

- bilibili <https://www.bilibili.com/>
- 优酷 <http://www.youku.com/>
- 官方论坛 <https://forums.terraria.org/index.php?forums/>
- 中文论坛 <https://www.bbstr.net/>

## 附录 人名代号

---

这个附录中包含了本书中提及的人名对应的论坛账号，以及本书写作过程中参考过的作者。

- putianyi888
  - bilibili: <https://space.bilibili.com/34937101>
  - 官方论坛: <https://forums.terraria.org/index.php?members/putianyi888.121300/>
  - 百度贴吧: <http://tieba.baidu.com/home/main?un=putianyi888>
  - 中文论坛: <https://www.bbstr.net/members/putianyi888.342/>
  - YouTube: <https://www.youtube.com/channel/UCsG1EimffDYWXBoZRekcMIA>
- usdanger
  - bilibili: <https://space.bilibili.com/34637318/>
  - 优酷: <http://i.youku.com/u/UMTcyMDA1MTY4>
  - 百度贴吧: [http://tieba.baidu.com/home/main?un=us\\_danger](http://tieba.baidu.com/home/main?un=us_danger)
  - YouTube: [https://www.youtube.com/channel/UCh\\_cLX4iAbM6tAIl0zu3elw](https://www.youtube.com/channel/UCh_cLX4iAbM6tAIl0zu3elw)
- dcfhft
  - bilibili: <https://space.bilibili.com/98605295/>
  - 百度贴吧: <http://tieba.baidu.com/home/main?un=dcfhft>
  - 官方论坛:
  - 中文论坛: <https://www.bbstr.net/members/dcfhft.135/>
- 屠宰场
  - bilibili: <https://space.bilibili.com/35610991/>
- 沉睡
  - bilibili: <https://space.bilibili.com/22871583/>
- 白霜心
  - bilibili: <https://space.bilibili.com/49886444/>
  - 优酷: <http://i.youku.com/u/UMTMyOTg1ODM4OA>
  - 百度贴吧: <http://tieba.baidu.com/home/main?un=>
- TNoName
  - bilibili: <https://space.bilibili.com/14462041/>
  - 中文论坛: <https://www.bbstr.net/members/tnoname.423/>
- 机锋哥
  - 优酷: <http://i.youku.com/u/UMjg3MTI2NDcwOA>
- ZeroGravitas
  - 官方论坛: <https://forums.terraria.org/index.php?members/zerograv>

itas.96/

- YouTube: <https://www.youtube.com/channel/UCyLQbVwYleCYzgl49dNAe0w>
- Programmatic
  - 官方论坛: <https://forums.terraria.org/index.php?members/programmatic.37545/>
  - YouTube: <https://www.youtube.com/channel/UCWGTyKTR5Kw3cDhyd2vIDew>
- TheRedStoneCrafter
  - 官方论坛: <https://forums.terraria.org/index.php?members/idkwhoiam129.57457/>
  - YouTube: <https://www.youtube.com/channel/UC9ekQo0v03BgFQsmGdWJpCQ>
- ekinator
  - 官方论坛: <https://forums.terraria.org/index.php?members/ekinator.61186/>
- DRKV
  - 官方论坛: <https://forums.terraria.org/index.php?members/drkv.67603/>
- DicemanX
  - 官方论坛: <https://forums.terraria.org/index.php?members/dicemanx.1706/>
  - YouTube: [https://www.youtube.com/channel/UC11YBm-\\_FbqWuI92o6zPXfw](https://www.youtube.com/channel/UC11YBm-_FbqWuI92o6zPXfw)
- MappyGaming
  - YouTube: <https://www.youtube.com/channel/UC-u6c7ppGML6icciJ0Q1oWQ>