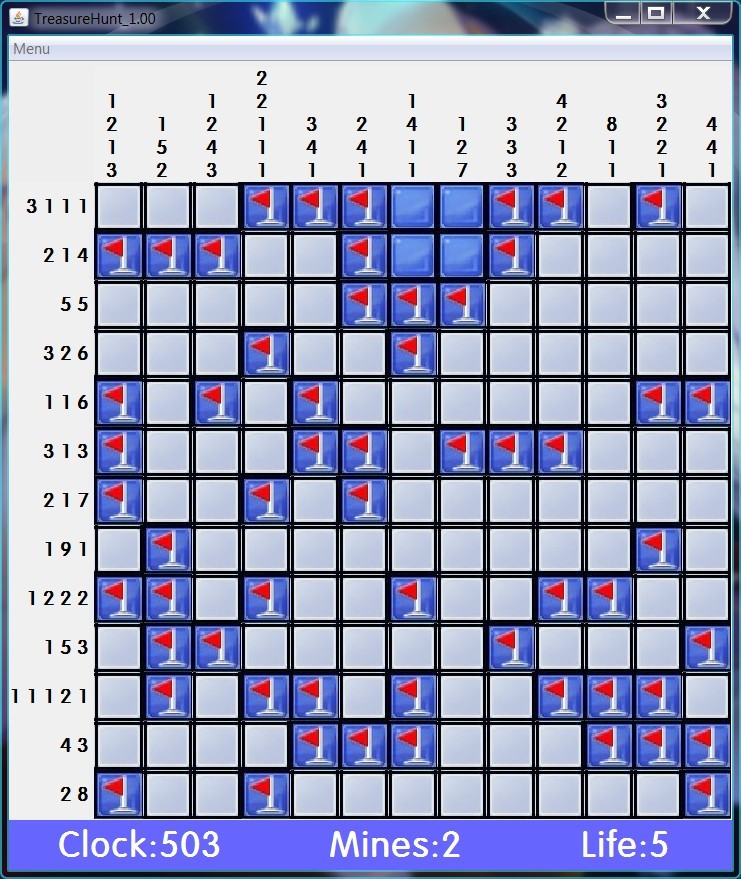
# TreasureHunt



一个类似扫雷的数字解密游戏。相似的界面，不一样的玩法，对玩家友好，也更为有趣。

目录

[TreasureHunt - 0 -](#_Toc486112057)

[Summary 1](#_Toc486112058)

[Background Story 1](#_Toc486112059)

[Points 1](#_Toc486112060)

[Dependencies 1](#_Toc486112061)

[References 2](#_Toc486112062)

[开发手记 2](#_Toc486112063)

[草创未就 2](#_Toc486112064)

[摸着石头过河 2](#_Toc486112065)

[拔云见日 3](#_Toc486112066)

[大功告成 4](#_Toc486112067)

## Summary

一个类似扫雷的数字解密游戏。相似的界面，不一样的玩法，对玩家友好，也更为有趣。

[GitHub](https://github.com/Sandigle/TreasureHunt)

**所有代码、素材均开源，转载请注明原作者：圣迪雅哥！**

如果想了解该软件的更多信息，或者报告Bug，可以联系编者。

邮箱：[492302909@qq.com](mailto:492302909@qq.com)

## Background Story

伟大的海贼王罗伊帕·凡·蒙特里安一生荣华富贵，他死后将所有财宝都藏在了一片深邃的大海中，并设下重重机关。落迫冒险家皮特·罗莉安为了振兴家族，穷尽毕生资源购置了搜寻装备，来到了这片不归地。究竟是海贼王的荣耀永存，还是冒险家的事业腾飞，就看玩家的选择。

## Points

* 目标：在有限的生命内找到地图上所有宝藏。
* 关键：处于左边和上边的冒险家装备显然不够格，它只能显示每行/列的宝藏分布情况。从左往右或自上而下，数字的值表示有几个宝藏连续排列，数字的个数表示有几段宝藏不连续排列。如“2 3 3”，表示该行分布了3段宝藏，从左往右分别是2/3/3个宝藏连续排列，每段宝藏之间至少有一个炸弹。
* 难度：在中低难度，几乎所有宝藏都可由提示唯一确定。而在高难度，宝藏一定无法由提示唯一确定。所幸冒险家福大命大，一次爆炸伤不到他的性命，他有足够的机会重新爬起。
* 评价：使用Logistic模型，目前只根据通关时间作为唯一标准进行评价。总体越快评价越高。

## Dependencies

运行程序至少装有Java Version 8 Update 101。

编译程序额外需要JMF 2.1.1e。

## References

所用贴图来自《win7扫雷》。

所用音效来自《洛奇mabinogi》。

## 开发手记

### 草创未就

接下来要做一件很酷的事情：编写一个小游戏。

就在昨天，看到一个名叫“Picross-Madness”的游戏。它是一个类似扫雷的逻辑推理游戏。第一眼看上去就被它所吸引，有趣的关卡设置和严密的逻辑推理让我爱罢不能。并且令人震惊的是，它竟然只有200KB。

在思考了它的运作原理后，发现它足够简单，让我自己实现。这便是工程的起因，以及接下来要达成的目标。

计划使用Java完成整个项目，因为对Java的理解和应用比C熟练了不少。这不代表我认为面对对象比面对过程要高了不少，从以往的经验讲，两者区别并不大。这个游戏的算法已经熟记于心，也容易实现，但问题在于交互界面的实现。在以往的开发中只实现过简单的GUI，但要用图像既能充当游戏内容，又要是交互的载体，就要另下功夫了。于是我决定先借鉴基于Java的扫雷程序。

### 摸着石头过河

我在“开源中国社区”上找到了由“haoguo”分享的Win7扫雷复制版。运行一看，基本能满足对图形操作的效果。接下来便要详细学习程序的每一步了。

开始不久就陷入了泥潭，几千行的程序几乎没有注解，要理清思路着实要费工夫。不仅如此，不标准的进位格式，大段的重复代码，非常低端的用拼音命名的习惯，不禁让人质疑作者的水平。而且我相信，这会是每一个程序员在接触开源代码时会遇到的问题。

到目前为止，理解程序的过程还算顺利。有实例又可以编译，使得学习变得如鱼得水，着实领会了多种新思路。特别是动态爆炸效果，代码简单效果出众，着实令人印象深刻。

但是在反复把玩程序之后，发现了程序的一连串问题。首先发现的就是，系统的计时功能失效。其次，在踩到雷时，本应该出现动态爆炸效果，但部分雷没有引爆。我修改了爆炸的进程，让它每次运行就使剩余雷数下降，并使雷在爆炸前先变成别的东西。结果，那些“哑雷”竟然根本没有运行爆炸进程，而剩雷数也顺利清零。并且，在反复运行后，界面上竟充满了“哑雷”。这一切都预示着一件事：上一局的雷被带到了下一局，新雷引爆了，但旧雷完好。回溯到重开游戏的代码，已经将所有雷都清空了，又没有发现任何生成雷的代码，可还是会有新雷生成。百思不得其解后，审视了一遍新雷生成的程序。这才发现，原来每一个雷区都设置了监听器，在点击后就会生成地雷。与此同时，计时功能的开关将被修改。但是该功能被放在了一个独立进程中，无法得到主进程的修改信息，这才解释了计时的失效。反复魔改后依然不能解决问题，对多进程的理解还需要进一步加深。

由此可见，一旦问题出现，其根源往往并不能直接发现。大多数人只能发现了表象，只治理了表象，结果就出现了“我们也没办法呀，只能把XXXXX”的悲剧。要真正解决问题，需要寻根究底，找出事物间的主要矛盾。敢于正视淋漓的事实，才是勇士，才有抱负的价值。

### 拔云见日

重新整理了程序的思路，把开关设为全局变量，并细化了各级开关，再修改了进程的开启位置，总算解决了计时和“哑雷”的问题。理解完源代码，总算可以开始开发自己的软件了。我将新游戏命名为“TreasureHunt(寻宝)”，因为它与“扫雷”既一脉相承，又与时俱进。

首先就要一脉相承。因为个人习惯，使用他人程序第一件要做的就是优化源代码。在删掉了多余功能和赘余代码后，代码量减少了一半左右。然后再重命名所有变量和函数，总算使代码看着清爽了不少。

然后则是与时俱进。“扫雷”使用周围的雷数提示，而“寻宝”使用每行/列的雷数提示。这部分的数值计算十分清晰，没花多少时间就实现了，而关键问题在于显示。显然，提示的数值可以用二维数组贮存，而数组的大小与雷区大小对应。自然，需要把数组按行/列显示在雷区的周围。那么这部分就需要2x2的容器用于显示雷区和提示。在对Java图形化界面的学习之后，发现边界布局只能存放5个既定位置容器；流式布局的初始状态只能显示一列容器；网格布局恰好就能实现2x2的布局，自然就是决定是它了。几番折腾之后，发现网格布局的每个容器大小一致，导致大量空间空白，非常难看，势必要被淘汰。几经周转，发现了网格包布局可以实现不同大小容器分布，这才实现了理想的布局。

布局解决后，需要将提示放进容器中。起初使用了JTextArea控件，但发现字体无法随窗口大小进行缩放，也难以设置行/列距使数字和网格对齐。此时开发陷入了僵局。但灵光一现，发现网格包布局居然放在了边界布局的中心位置，这才认识到布局原来可以嵌套使用。于是在2x2布局中加入了由JLable控件组成网格布局，这才完成了数值与雷区的一一对应。

至此，游戏已经可以正常进行了，剩下的就Manuel和交互效果的设计了。

由此可见，事物的发展是螺旋上升的，但道路是曲折的、迂回的。每一次的探索，也许不会马上发挥效果，但总有其存在的的价值。过去所做所为造就了如今的结果。真理有时会迟到，但绝不会缺席。

### 大功告成

开始着手用户体验的提升，这部分需要的是微小而重要的开发。它通常无法让人直接看出来，但在实际操作决定着软件的使用的效果。

如何能让用户从游戏中获得快感而持续的进行游戏了？丰富每一次行动带来的反馈是至关重要的。贴图的变化、立体阴影效果、震撼人心的音效，这些都让用户身临其境，从鼠标的点击中产生实际“挖宝”的感觉。这一说这些反馈赋予了游戏意义，是不同游戏间的明确差距。这次选用了情怀游戏《洛奇》中的部分音效，相信部分有相同情怀的玩家在听到失败音效时会心头一惊吧。(｀^ω^)

术业有专攻，不同玩家对游戏的理解显然不同，那么不同难度的设置就体现了人性化。中等难度可以仅靠逻辑完美通关，高级难度还需要对模糊状态的最优分析，而初级难度几乎没有难度。适当的难度划分显然是不同层级玩家的专属优化。

抽象化玩家的表现，建立合适的评价指标，给玩家一个明确的努力方向。事物发展趋于收敛，而中庸者居多，那么用Logistic模型建立评价系统就再好不过了。

此外还有边框的大小、窗口的缩放、字体的选择以及不同屏幕下的测试。虽然还有一些难以解决的bug，但对于一个小游戏而言，这些已经足够了。