**BOMB!  
——基于汇编语言的益智游戏**

***Version 1.0***

**编写者**

**Prepared by**

**团队：王陈陈**

**Group Name: wcc**

|  |  |  |
| --- | --- | --- |
| **王肖佑** | **2011013273** | **wxyyxw09090@sina.com** |
| **陈璐** | **2011013249** | **chenlu11@mails.tsinghua.edu.cn** |
| **陈华榕** | **2011013236** | **chenhuarongzp@gmail.com** |
|  |  |  |
|  |  |  |
|  |  |  |

|  |  |
| --- | --- |
| **Instructor:** | **王朝坤** |
| **Course:** | **计算机网络体系结构（2）** |
| **Teaching Assistant:** | **陈俊** |
| **Date:** | **2014/04/09** |

目录

[文档修订 iii](#_Toc384890628)

[1 开发概况 1](#_Toc384890629)

[1.1 开发团队 1](#_Toc384890630)

[1.2 开发时间轴 1](#_Toc384890631)

[1.3 开发成果 1](#_Toc384890632)

[2 游戏规则 1](#_Toc384890633)

[2.1 游戏背景 1](#_Toc384890634)

[2.2 操作方式 2](#_Toc384890635)

[2.3 操作规则 2](#_Toc384890636)

[2.4 胜利条件 2](#_Toc384890637)

[2.5 失败条件 2](#_Toc384890638)

[3 程序模块 2](#_Toc384890639)

[3.1 Win32入口 2](#_Toc384890640)

[3.2 数据与逻辑 3](#_Toc384890641)

[3.3 文件存取 4](#_Toc384890642)

[4 总结 4](#_Toc384890643)

# 文档修订

|  |  |  |  |
| --- | --- | --- | --- |
| 版本号  Version | 主要作者  Primary Author(s) | 简述  Description of Version | 完成时间  Date Completed |
| 1.0 | 王肖佑、陈璐、陈华榕 | 本文档撰写完成。 | 2014/04/09 |

# 开发概况

## 开发团队

【肖佑】介绍团队成员，可用幽默的语言。

## 开发时间轴

【nunu】简单说说开发进度与时间的关系，简单说就是表明我们启动早、肯投入、效率高。

## 开发成果

我们经过这段时间的开发，设计并用汇编语言在Windows平台实现了一款名为BOMB!的游戏。

该游戏基于Win32汇编，操作简单，规则有趣，具有较高的可玩性，我们开发人员在测试过程中发现，这个游戏一玩就“根本停不下来”。

# 游戏规则

## 游戏背景

火药由中国人发明，距今已有超过一千年了。最近几年由于科技进步，人类掌握了通过信息技术合成炸弹的方法，若能推广使用，将大大减少炸弹制作过程中的资源消耗以及引爆后的环境污染。

这看似不可能的研究成果正是在清华大学软件学院某实验室诞生的，但由于该技术严重影响火药生产商的利益，在推广过程中受到了很大阻碍（也因为此，现在甚至连软院的同学都几乎不知道这个能震惊世界的成果）。该实验室希望通过事实证明这项技术是切实可行、可靠的，然后将该成果公之于众。你被该实验室招募为一名终身测试人员，将参与未来对炸弹合成技术的测试。

WCC团队在该实验室的委托下，开发了BOMB!游戏，以此模拟炸弹合成、引爆等过程，并加入游戏化元素。这样，测试人员只需要通过玩BOMB!游戏就能在悄然间完成一次次的测试工作。

您可以将BOMB!游戏分享给亲朋好友，那将大大加快测试进度，但请切记不要将涉及的炸弹合成技术测试相关背景透露给任何人。为了保证您的人身安全，也请不要透露您是该技术的终身测试人员这一身份。

由于涉及的逻辑分支非常之多而且必须通过人力驱动，因此我们初步推算，即使全球能有10亿人不间断地参与，该实验室要完成完整的测试也大约要到2222年。

## 操作方式

【nunu】介绍鼠标、键盘在本游戏中的作用。说得简单些，大概就是哪些键用到了，一句话说明用来干嘛，比如方向键只需要介绍是将所有方块整体向某方向移动，具体移动规则在下一节说。

## 操作规则

【nunu】说明游戏的核心规则。让人看完能明白怎么玩，能理解所有游戏元素与游戏表现即可。

### 游戏元素

墙、炸弹、数字、空白区域等。

### 变化规则

按方向键后的移动规则、数字的合成规则、炸弹的合成规则、炸弹的爆炸效果等

## 胜利条件

【nunu】

## 失败条件

【nunu】

# 程序模块

这里可以说明各个模块的具体实现。可根据需要调整各级标题。

## Win32入口

【肖佑和nunu】

### 程序入口

### 界面交互

### 附加交互

比如音乐什么的，除界面视觉交互外的其他交互。

### 游戏动画

### 按键队列

## 数据与逻辑

这部分主要涉及bomb\_core.inc和bomb\_core.asm两个源代码文件。

### 地图与基本数据

|  |  |  |
| --- | --- | --- |
| 标识名 | 类型 | 说明 |
| map | 行列数相等的SDWORD类型的二维数组。 | 地图，可能取值负数表示墙，0表示空地，还可取值2^n（n>=1）表示用于合成的数字。 |
| mapSize | SDWORD类型的变量。 | 设置的地图大小，取值必须满足0<mapSize<=20。 |
| bombTarget | DWORD类型的变量。 | 当合成出>=bombTarget的数字时，将得到炸弹，同时bombTarget<<=1。地图初始化时将设定为4。 |
| resultQueue | 一维DWORD类型的数组。 | 用于给交互界面返回结果的队列，详见下一节“结果队列”。 |
| emptyBlocks | SDWORD类型的变量。 | 表示当前空地的格子的数量。 |
| numberBlocks | SDWORD类型的变量。 | 表示当前数字的格子的数量。 |

### 结果队列

为方便播放动画，当用户请求执行某方向动作后，数据逻辑部分通过结果队列将动画详情返回给交互界面调用者。结果队列是一维DWORD类型的数组，其特定二进制位有特别含义，每个DWORD表示一个结果。

以下所称第x位代表二进制的右起第x位，以a\_n表示数a的第n位，以a\_m\_n表示数a的第m到n位。如对于b=10010b，有b\_1=0，b\_2=1。

结果队列有四种可能：地图块移动、数字合成、出现炸弹。

#### 地图块移动

以a表示要加入结果队列的数，记要移动第x行第y列的地图块，移动距离为z。这里不需要记录移动方向，因为结果队列一定是在执行某方向移动后得到的，方向已经隐含在其中了。

则a\_2\_1=0，a\_7\_3=x，a\_12\_8=y，a\_32\_13=z。

#### 数字合成

数字合成的动画在地图块移动的动画完成后进行，只需要指明哪个地图块进行了合成即可，假设为第x行第y列的地图块，并以a表示要加入结果队列的数。

则a\_2\_1=1，a\_7\_3=x，a\_12\_8=y，a\_32\_13=0。

#### 出现炸弹

炸弹出现的动画涉及炸弹的位置，需要指明的是地图块的行x和列y。以a表示要加入结果队列的数，则a\_2\_1=2，a\_7\_3=x，a\_12\_8=y，a\_32\_13=0。

爆炸后应将周围的墙夷为平地，这是隐含的，不需要在结果队列中体现。

### 方向逻辑

#### DoMove

通过DoMove过程实现方向的操作。该过程只有一个参数，即方向，可取值DIR\_UP、DIR\_DOWN、DIR\_LEFT、DIR\_RIGHT，通过eax传入。

在其中针对不同方向进行方块移动，依照游戏规则，移动的逻辑顺序是：根据移动方向的反方向枚举各个地图块（比如向下移动就列数从大到小枚举，向上移动则列数从小到大枚举），每次找到不为墙的地图块A，就向移动方向找到第一个不为0的地图块B，若B不为墙，就接着判断A，如果A为0，就直接把B移至A，而如果A=B，就将两个地图块合成，其他情况不操作。

当地图上的所有地图块都经过上述移动操作后，再进行合成炸弹判断，枚举一遍地图，将所有满足合成条件（数字>=bombTarget）的地图块替换为炸弹，然后将该地图块四周的墙炸为空地。

上述所有有效操作均会在结果队列产生新内容。

该过程返回结果队列的长度，通过eax返回。

#### AddNum

该过程调用后将给地图增加一个数字块，即用一个数字替换一块空地。

首先根据当前空地数生成一个随机数X，然后根据该随机数按特定顺序找到地图上第X块空地。再通过随机数决定要生成的数字，90%概率生成2，同时10%概率生成4。

该过程通过eax返回生成的情况。以a\_n表示返回值的第n位，a\_m\_n表示返回值的第m到n位，则返回值表示在第a\_11\_7行、第a\_6\_2列生成了一个数(2<<a\_1)，这里的行列数可取0。

### CheckMap

该过程调用后检查地图状态，返回值有三种可能：MAP\_CONTINUE表示可继续游戏；MAP\_WIN表示已将所有墙夷为平地，游戏已取得胜利；MAP\_FAIL表示已没有可移动数字块，游戏已经失败。

## 文件存取

这部分主要涉及serialization.inc和serialization.asm两个源代码文件。

### 序列化

有必要保存的关键数据是：地图大小mapSize、地图map、合成目标bombTarget。将它们按顺序以二进制编码即可完成序列化。同样能方便地进行解序列化。

这里map将保存整个二维数组（而非只保存mapSize大小的地图），这样实现起来更简单，且给保存文件加入冗余信息，也是数据的一种保护。

### 文件保存

SaveSerialization过程实现文件保存，会按序列化顺序将二进制数据输出至工作目录下的bomb.wcc文件。

### 文件读取

LoadSerialization过程实现文件读取，会按序列化顺序将二进制数据从工作目录下bomb.wcc文件中读入。

# 总结

也不知道是谁规定的，实验文档通常有一个总结。

BOMB!从规则到实现都包含我们的耐心思考与劳动，虽然从操作方式、界面上能看到诸多2048游戏（<http://gabrielecirulli.github.io/2048/>）的影子，也都给用户带来“根本停不下来”的感觉，但我们针对规则的改进让游戏更具趣味性，同时引入的5\*5地图如果按常规方式很难取胜（因为理论上来说需要消耗大量时间），但通过我们特别设定的小trick（比如同时合成多个炸弹只将bombTarget翻倍一次）可以大大缩短取胜的用时。

实际上，BOMB!对游戏者控制特定地图块的能力有了更高的要求，游戏者应具备将特定地图块移动至特地新位置的能力，否则将造成大量无用的合成，进而导致陷入时间的无底洞。

总之，我们对BOMB!的游戏设定、汇编语言的实现效果都非常满意。