****

**本科生毕业论文（设计）**

题 目："Dyscrabble"动态拼字游戏的设计与实现

院 系： 软件学院

专 业： 软件工程（嵌入式软件与系统）

学生姓名： 赖卓航

学 号： 11331157

指导教师： 刘宁 （副教授）

二〇一 五 年 三 月

**摘 要**

拼字游戏是一款起源于西方国家的文字类益智游戏，深受许多玩家喜爱。现有的大多数拼字游戏都是从指定词典取词后生成拼字地图并给予玩家少许提示以完成拼图。本次设计主要针对传统的拼字游戏进行形式上的创新，增加了从网络中下载实时新闻、学术类文章，让玩家阅读后从文章中选词填入等功能。选取的均为词频较低或较为生僻的词汇。

论文首先介绍拼字游戏的发展历程。再介绍游戏界面的设计与制作过程。然后介绍游戏各个模块的设计过程，包括通过网络爬虫技术设计模块从指定网站上获取英文文章，通过词频数据库分析文章词汇并得出词频最低的若干词汇，以及最后基于选取的词汇通过拼字地图生成算法生成地图等等。接着介绍游戏中使用到了算法以及优化策略。最后将展示这款动态拼字游戏，从画面中感受其带来的游戏效果。

摘要内容应概括地反映出本论文的主要内容，主要说明本论文的研究目的、内容、方法、成果和结论。

**关键词： 网络爬虫 词频统计 地图 优化**

ABSTRACT

Scrabble game is a type of popular literal puzzle game originated from the west. Currently most of the scrabble games pick up words from a specified dictionary, use them to generate a scrabble map for players and provide with several tips. Based on this traditional design model, I would like to make some innovations, such as downloading news or academic articles automatically from the websites, words picking from these articles according to their statistic frequencies. Also for players, their task is not remembering the words in the dictionary, but reading and paying attention to those long and uncommon words. Thanks to the advanced and intelligent web crawler technique, lots of plain texts can be scratched back with high efficiency for analysis.

**Keywords: web crawler, frequency statistic, map, optimization**

目 录

第一章 概述/引言 3

1.1 XXX问题的背景和意义 4

1.2 XXX问题的描述 4

1.3 本文的工作 4

1.4 论文结构简介 4

第二章 XXX综述 5

2.1 ××× 5

2.2 ××××××× 6

2.3 ××× 6

2.3.1 ××× 6

第三章 提出的×××方法 7

3.1 ×××问题描述 7

3.1.1 ×××××× 7

3.1.2 ×××××× 7

3.2 ××××方法 7

3.2.1 ×××××× 7

3.2.2 ×××××× 7

第四章 ×××算法 8

4.1 ×××××× 8

4.1.1 ××× 8

第五章 仿真/实验结果与分析 9

5.1 ×××××× 9

5.1.1 ××× 9

第六章 总结与展望 10

6.1 ×××××× 10

6.1.1 ××× 10

参考文献 11

相关的科研成果目录 13

致 谢 14

附 录 15

# 概述

## 拼字游戏创新的背景和意义

随着计算机技术的不断发展，计算机俨然已成为人们生活中不可或缺的元素。现今人们在精神上的追求越来越高，对游戏的种类和玩法的创新需求呼之欲出。依靠计算机硬件和软件的迅猛发展，计算机平台上的游戏的高速发展也正迎合了人们追求新颖、刺激、流畅的游戏效果的心态。具体地说，当前计算机平台上的游戏种类多样化程度越来越高，游戏的效果愈加炫丽多彩，游戏操作的流畅性也得到了大幅度的提高。许多历史悠久的游戏通过计算机平台，由实体型转变成了虚拟型。这些游戏的基本形式得到更加生动的表示，并且随着时间的迁移，人们不断在已有的游戏基础形式上派生出其他的新颖的形式与玩法，使得这些古老的游戏能够历久弥新，继续传承下去。

拼字游戏在1938年由建筑师Alfred Mosher Butts设计出来，是一款文字型益智游戏。根据拼字游戏最初的设计，玩家需要从指定的字典中选取符合要求的单词填入拼字地图格中并满足一定要求。一直以来，拼字游戏以其对玩家英语技能的锻炼以及其趣味性广受人们的好评。具体而言，传统的拼字游戏对玩家的单词记忆、联想能力进行了考验。而现代的创新意识驱使着人们对这类传统文字游戏进行改进和创新，使其在游戏流畅性、趣味性、综合性上日渐提高。而本次设计，即针对传统的拼字游戏在玩法上进行创新，融入了即时文章源下载的思想，不仅锻炼了玩家的联想能力，也锻炼了玩家的阅读能力，可谓一石二鸟。

由于计算机平台的多样性，大部分计算机语言只能通过在不同的平台上使用不同的代码来设计游戏。这使得一个计算机游戏拥有跨平台性的成本大大增加，对游戏的多平台化发展造成了影响。而Java语言与生俱来的跨平台性却很好地解决了这个问题。Java语言从1991年创建到现在，已经逐渐变得成熟，在越来越多的软件设计需求中发挥其优势。Java的Swing套件使用纯Java语言编写，在Java虚拟机中运行效率高，已经成为许多中小型游戏开发者的选择。虽然Java语言并不是为游戏编程而诞生，但是其跨平台、高性能等优点已经能在游戏编程中得到充分体现。

## 游戏设计难点描述

拼字游戏作为一种传统的文字类游戏，在玩法上已然被广大玩家所知晓。游戏的操作简单，然而在游戏的过程中，一些玩家由于自身水平的限制，常常受困于游戏难题中许久不能得到答案，使得传统拼字游戏的流畅性不如其他类型的游戏（如战略竞技类游戏）。如此，拼字游戏给玩家带来的能力的提升受到了限制，其游戏体验也并不能吸引到大量的玩家。因此，本次设计主要针对传统拼字游戏玩法较为古老且流畅性低的特点，希望对其进行玩法上的改进和创新。在对玩家英语技能的锻炼方面，希望能增加英语快速阅读与记忆的能力，并减少漫无目的的单词联想行为所占的比例，提升玩家的答题速度，进而增加游戏的流畅性和紧张性。

在游戏形式上，本次设计的动态拼字游戏"Dyscrabble"将从网络中提取实时性文章，分析文章的词汇并筛选出符合要求的若干词汇。进而针对这些词汇为玩家玩成待填的拼字地图。玩家需要通过阅读文章，对文章中较为生僻的词汇进行记忆，以便在填字流程中快速地填入词汇。

在技术实现上，可以将Dyscrabble分为三部分：在线新闻类文章的抓取与下载、文章的词汇词频分析与筛选、填字地图的生成。对于文章的抓取，如今网络上有许多的成型的网络爬虫可供参考。也有专门的网络爬虫生成框架，可以快速地为开发者定制针对特定网站的网络爬虫。本次设计的网络爬虫部分采用Python语言实现，未采用任何已有的框架。在网络爬虫的开发过程中，如何为特定网站定制爬虫对该网站的URL进行提取和筛选以及对HTML文档中有用信息的筛选成为难点。这需要设计者充分了解该网站的服务器中网页文件的组织形式，以及这些网页文件内容的构成与排序。

对于词汇词频的分析筛选，现有的许多功能强大的单机数据库依靠其高度的存取优化技术为分析筛选操作的高效性提供了有力的支持。然而在实际开发中设计者意识到，由于文章中词汇的数量较大，且不少高频词汇在文章中频频出现，频繁的数据库存取操作使得操作性能不高。因此该部分的难点在于如何设计一个合理的缓存机制，减少系统访问数据库的次数，提高词频分析的效率。

而在地图生成过程中，系统需要合理地规划地图中每个单词的存放位置，使得地图中的单词个数最多，生成的地图复杂程度最高。同时为了增加游戏的不确定性，生成的地图需要具有一定的随机性，使得一篇文章能对应生成多种不同的地图，解决从网络中获取文章资源的不足问题。具体地说，如何在词汇选取、词汇排放的策略中融入随机化的思想也是设计地图生成算法时需要思考的问题。

## 本文的工作

针对1.1中对Dyscrabble动态拼字游戏设计目的的描述以及1.2中对设计过程中对实现难点的分析，设计者对于不同的模块依次采用了如下方案进行设计与开发：

对于网络爬虫模块，设计中会使用Python语言进行爬虫开发。特别地，Python中的网络访问动态库对网络的访问速度、解码速度与准确性进行了提升，如urllib2、HTMLParser等。在网页文件的分析与有用信息的筛选中，Python的正则表达式模块re起到关键作用。re模块通过订制需要获取的信息的正则表达式形式，快速地在HTML文件中寻找符合正则表达式形式的内容。同时，正则表达式对已获取信息的进一步加工与筛选也起到了很大的作用。在解码方面，Python的decode、encode函数能够方便高效地解码网络内容。网络中许多英文网站为了兼容英语以及一些西语字符，多采用iso-8859字符集。但由于网站开发者的目的与水平并不一致，因此并不是每个英文网站都采用同种编码。甚至有的网站网页文件采用的编码字符集与网页属性栏中显示的字符集不一致，由此造成解码的困难。而HTMLParser模块在HTML文本信息的提取、解码方面首屈一指，能有效解决这种问题。

由于系统主程序使用Java语言进行开发，而网络爬虫模块使用Python脚本语言进行设计，在模块衔接过程中涉及到Java程序对Python脚本的调用。目前有许多Java平台上专门为Python调用而设计的jar包，如jython。这些包提供了API使得设计者在Java平台上能够方便地调用Python的函数。但由于这些jar包对Python的第三方库的支持存在缺陷，经过衡量后设计者选择使用Java的Process类（进程类）对Python脚本进行调用。

在词频统计模块中，设计者采用了轻便小巧的单机型数据库sqlite3进行词汇词频信息的存放。通过在数据堂中下载由北京语言大学——汉语国际教育技术研发中心整理的从华尔街日报语料库中统计的拥有20万常用单词及其变形的绝对词频的词频库作为系统的初始词频库。采用Java中的HashMap数据结构构建内存缓存区，对已访问的数据项进行缓存和更新。并在分析完成后将更新的词频从新写入到数据库中。

设计者使用重复的广度优先搜素算法（BFS）以生成复杂的地图。在BFS过程中，为了减少无用节点进入队列的次数，在每次新节点入队过程中都会通过对其周围节点的分析与规律的寻找判断其加入队列的必要性。生成的地图以其填入单词个数、横纵单词交汇点（该点在横向、纵向分别为两个词的字母）的个数以及生成地图的面积大小作为衡量其质量的因素。设计者通过对三种因素的重要性进行排序，设计出地图质量的计算公式，由此算出每次生成地图的质量分数，取最高分者作为本次地图生成算法的输出地图。

最后，在程序GUI的设计上，设计者使用了Java的Swing工具包进行组件开发。通过Java的内存垃圾自动收集机制，较好地实现了在游戏运行过程中由于页面跳转、功能限制等情况出现的内存中组件资源的弃置与内存回收操作。

## 论文结构

本文后续部分将依次展示以下内容：第二章将介绍Java的GUI组件开发包Swing，并与其他Java的GUI开发包进行性能上的比较，接着介绍网络爬虫的架构与工作流程、Java中Process类调用Python脚本的过程，以及sqlite3数据库与Java中HashMap数据结构分别对数据存取的原理与效率比较；第三章将从宏观上概括Dyscrabble的系统架构和模块划分；第四章将详细描述在每个模块设计过程中涉及的以及设计者提出的设计与优化方案；第五章将描述游戏GUI的设计以及展示部分界面；第六章将描述本次设计的结论以及设计者的展望；第七章为论文参考文献。

# 相关工作介绍

本章将介绍Java游戏开发过程中GUI设计、数据库存取优化、脚本语言的调用方法、网络爬虫的设计理念以及其已有的相关成果。

## Java Swing GUI组件开发包

Swing是Java的GUI组件开发包，它以抽象窗口工具包（AWT）为设计基础。Abstract Windows Toolkit(AWT)为Java平台上最早的GUI组件开发包，Java的每个版本都对其完全支持。AWT依靠Java GUI程序运行的当前平台的系统函数调用来实现其功能，如：AWT的图形函数与当前平台操作系统的图形函数存在一一映射的关系。为了秉承Jave的平台无关性，AWT函数调用需要顾及到每一个操作系统的底层函数。因此其能实现的功能较少，为各种操作系统所支持的操作的交集，遵循“最大公约数”原则。故AWT控件也被称为重量级控件。而Swing拥有AWT的所有功能，并且对AWT的功能进行了扩充。Swing工具包完全采用Java语言设计，因此其具有较强的跨平台性。对于AWT没有的控件功能，Swing利用AWT中的基础绘图方法对控件进行模拟。Swing也被称为轻量级控件。

相比较而言，由于AWT直接调用系统函数进行处理，其运行效率更高一些。但由于如今大部分计算机的硬件配置水平使得由于Swing在运行效率上的相对低下而造成的性能损失较少，因此两者的性能差已经不能成为选择的考虑因素了。而在实现组件的功能上，Swing远胜于AWT。这也是设计中使用Swing组件进行开发的原因所在。

另外，Standard Windows Toolkit(SWT)也是Java GUI工具包之一。其设计的初衷也是扩展AWT的功能，并且在跨平台特性上有更好的扩展。对于某些只在某个操作系统上特有的组件，SWT会使用模拟的方法在其他操作系统中实现相应的组件功能。因此，SWT遵循“最小公倍数”原则。SWT的模拟方法不同于Swing，它仍会调用一些系统底层函数作为支撑。且由于现在的SWT版本还无法完全模拟一些组件的特征，故SWT的组件集拓展仍稍逊Swing一筹。但是，SWT的内存消耗相对较少。

总结来说，考虑工具包提供的组件功能数目，AWT提供最少的组件功能，SWT次之，Swing是最多的。

## Java HashMap与sqlite

Jave HashMap是Java语言的映射(Map)结构中的一种，是基于哈希结构的Map接口的实现。在Java中，所有的对象都从Object类中继承了函数hasCode()，hasCode()将会以整数形式返回该对象在内存中的地址，并且将整数映射到当前HashMap容器中的特定位置。因此，hasCode()作为哈希函数，尽量保证了不同的对象拥有不同的哈希值。在HashMap中存储的每一个对象都是一个Entry<K,V>类的对象，即键值对的形式。在程序运行过程中，创建的HashMap对象存储在程序内存空间的堆中。以键为参数查询该键在HashMap中对应的值的操作耗时约为1ms，是相当高效的。

sqlite是一款轻量级关系型数据库管理系统。与大多数数据库管理系统不同的是，sqlite并不遵循Client-Server的模式。它嵌入在本地系统中，服务于本地数据的存取。在存取数据方面，sqlite如同其他大型DBMS一样，支持事务的ACID性质，而其查询语言也采用标准的SQL查询语言。sqlite为多种程序语言提供了便捷高效的API，使其可直接被以不同编程语言编写的程序访问。sqlite自身也设有缓冲区，使用LRU（最近最少使用）的缓存替换算法，减少访问数据库的次数。通过实验我们知道，由于受到访问sqlite数据库的Java的接口函数的运行时间影响，对数据库的存取时间约为38ms。若数据库缓冲区中存在，则存取时间可降低到7ms。

## Java Process的脚本调用

Process类是Java语言自带的抽象类，其中封装了一个进程。Process类提供了在Java代码中对另一个进程进行执行、等待、销毁、检查进程执行返回值等操作。通过使用Jave的Runtime类中的Runtime.exec()函数可创建一个子进程并可赋给Process类的对象。创建子进程的命令与命令行或终端中运行程序的命令一致。在这种调用方式下，子进程的结果并不会在命令行或终端中显示。相反，其标准输入流、标准输出流、标准错误流都将重定向到父进程中。在Process类中分别使用getInputStream()、getOutputStream()、getErrorStream()可获取子进程的流。在设计中，主程序需要执行使用Python语言开发的网络爬虫进程Crawler.py，并实时读取当前爬虫抓取的网页URL。Process类能够较好地满足设计的需求。

另外，Jython（原为JPython）是一个使用Java语言编写的Python解释器。Jython使用Java本身的模块对python代码进行解释与处理。然而Jython并不支持Python中依赖于C语言的模块，如在设计中使用到的python的sys模块。因此此设计并不采用Jython解释器作为调用Python程序的工具。

## 网络爬虫

网络爬虫是一种依据一定的规则，自动化抓取网页文件中的有用信息的程序。随着网络技术的不断发展与革新，万维网上拥有的信息量正爆炸式地迅速增长。最初，网络爬虫的形成是为了解决搜索引擎 对万维网海量信息的自动获取问题。原本的网络爬虫设计目标在于尽可能地覆盖万维网中的信息，因此也被称为通用型网络爬虫。后来由于搜索引擎服务器的限制以及用户的需求，派生出了仅聚焦于某个信息领域的聚焦型网络爬虫。这类爬虫能定向抓取某个领域或者具体到某个网站中的信息。通用型网络爬虫需要处理大量内容与形式皆不相同的网页文本，故仅针对所有的网页HTML文本的共同点处理，其针对单一领域或网站的信息处理能力便成为了短板。相反，聚焦型网络爬虫能针对单一领域获网站的HTML文件与文本内容的组织形式，进行详尽搜索与精确定位。本次设计完成的是对香港英文虎报的官方网站（www.thestandard.com.hk）中即时新闻内容的抓取与信息过滤的聚焦型网络爬虫。

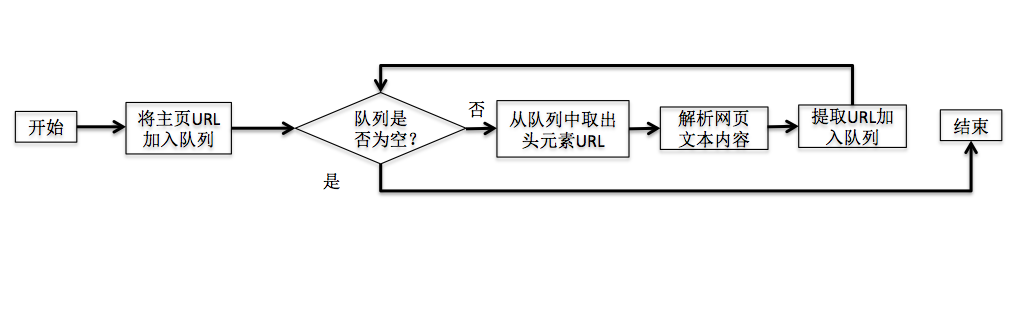
聚焦型网络爬虫在设计上需要思考如下问题：

**URL搜索的策略与算法：**URL搜索算法可分为三种：广度优先搜索（BFS）、深度优先搜索（DFS）以及最佳优先算法。

**在网页中对所需数据内容的定义：**定义所需的数据结构，如文本、图片链接、关键词等。

**对网页内容的过滤与分析：**得到了需要的数据后，数据中仍然可能出现残余的HTML代码。需要使用正则表达式将无关的代码过滤。

若采用BFS算法作为URL的搜索策略，则聚焦型网络爬虫的工作流程如图所示：



首先，网络爬虫将从初始网页URL开始搜索，分析网页内容，并将初始网页文本中符合要求的URL加入到队列中。随后采用BFS算法不断取出队列中的URL元素，分析其该URL对应的网页的内容，抓取出有用的信息并继续添加该页面的URL到队列中。定义在当前网页中点击某URL进入新网页的操作为一次跳转。则与初始网页之间跳转数越少的网页与初始网页的内容联系相对更为紧密。一般来说，使用BFS算法使得爬虫能够快速地在离初始网页跳转数越少的网页中获取内容。若采用深度优先搜索（DFS）的方式则可能使得抓取的网页信息价值变低。

# 总体设计

软件的总体设计一般分为两部分：系统功能设计与模块划分。系统功能设计主要列举系统的功能并确定系统的实现方案；模块划分主要在实现方案确定后将系统划分为不同的模块分别进行设计。

## 系统功能设计

Dyscrabble动态拼字游戏是一款单人游戏。对于玩家来说，打开游戏画面后，玩家可点击"start"按钮，进入难度选择。难度分为"Easy"、"Medium"、"Hard"三种。当玩家选择其中一种难度后，将被告知是否确定开始游戏。当确定开始后，将能看到加载完成的主界面。主界面分为四部分：第一部分为文章显示区，显示目前需要玩家阅读的新闻。第二部分为填字地图，上面的填字格或不可填入字母，或已填入不可修改的提示字母，或为空白等到玩家填入字母。第三部分为控制按钮区，玩家可以在按钮区中点击"Finish"玩家本次填词。也可点击"Back"结束游戏并返回开始界面。当点击"Finish"按钮后，玩家将被告知所填答案是否正确，并在填字格区中看到单词正确与错误的颜色标识。完成一轮游戏后，点击"Next"进入下一轮游戏。第四部分为计时器，显示当前轮中玩家尚有的作答时间。

开始游戏后，玩家首先需要阅读文章显示区中的新闻，记忆文章中长度大于4的单词。然后将自己看到的词正确地填入到填字地图中，使得填入的字母在横、纵方向满足单词的正确拼写。填字地图中所有的单词均来自文章，但文章中并不是所有的词都被纳入到填字地图中。玩家在填字的过程中需要不断回忆或查找在文章中读到的单词，从而选出正确的单词填入。地图生成过程中会根据玩家所选的难度，适当给出部分格子显示为橙色的正确字母，以提示并限制玩家的输入。

点击"Finish"提交答案后，填字地图中所有的字母颜色会发生改变。绿色的字母表示填入正确，而红色的字母表示填入错误。

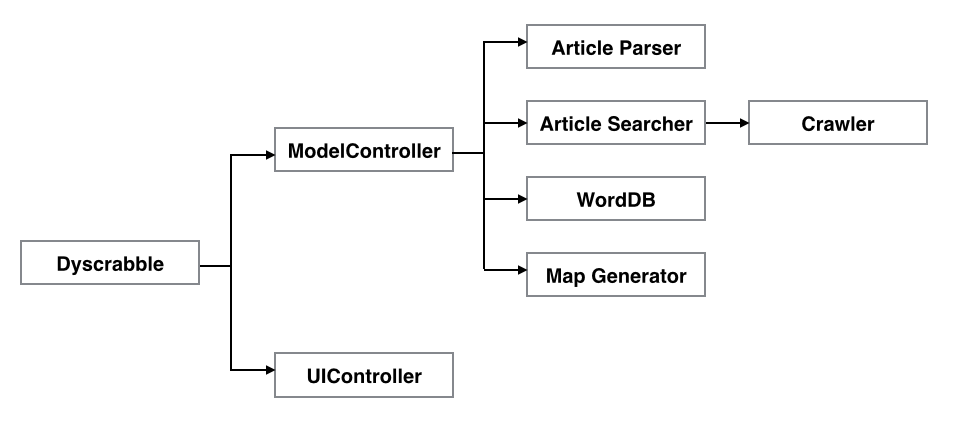
对于系统来说，在玩家打开游戏界面后，系统需要测试当前计算机的网络状况及网速。当网络运行良好时，会开启网络爬虫对英文虎报官网上的最近新闻进行抓取。抓取文章数目取决于当前网速，以保证文章加载时间不会太长。若网络不通，则关闭爬虫模块，并选择启用本地储存的文章作为源素材。当玩家选择难度并开始游戏后，系统将从本地中寻找可用的文章，并以不同的概率选择不同日期的文章。（时间越久远的文章选取到的概率越低）然后对所选文章进行词频分析，选择长度大于4的单词中词频最低的20个词，并随机抽取至多15个词使用地图生成算法生成地图并显示在填字地图区中。同时，计时模块打开，显示倒计时。当玩家点击"Finish"后，系统通过检测用户在填字地图中填入的字母组成的单词是否在文章中出现以决定玩家的输赢。当玩家点击"Next"按钮后，文章显示区、填字地图区以及计时器将重置，进入新一轮的游戏。

由于系统同时需要在前端显示游戏内容并响应玩家的操作和后台的数据处理，因此多线程的使用十分频繁。网络检测、词频分析、地图生成与计时模块均采用多线程的形式。而网络爬虫更是采用了子进程的形式，确保主线程能及时响应玩家的操作。

## 模块划分

在对系统的功能进行描述与分析，并确定解决方案后，需要将系统划分为若干模块，分模块进行设计开发。最后将各模块合并，使其协同工作。

Dyscrabble的系统可分为UI界面控制器以及游戏模型控制器两部分。UI界面控制器控制开始界面与游戏界面的显示与响应；模型控制器又可继续细分为：文章分析器、文章搜寻器、地图生成器、模块组控制器、网络爬虫以及存储操作器。各功能模块图示如下：



各模型控制器功能介绍：

**Dyscrabble.Models.ArticleParser（文章分析器）**：其中包含了对文章中含有的非英文字母的过滤、词频获取函数和单词选择函数。文章分析器在获取词频时会调用存储操作器对数据库进行访问。

**Dyscrabble.Models.ArticleSearcher（文章搜寻器）**：其中包含对网络爬虫的调用以及对本地文章的抽取函数。

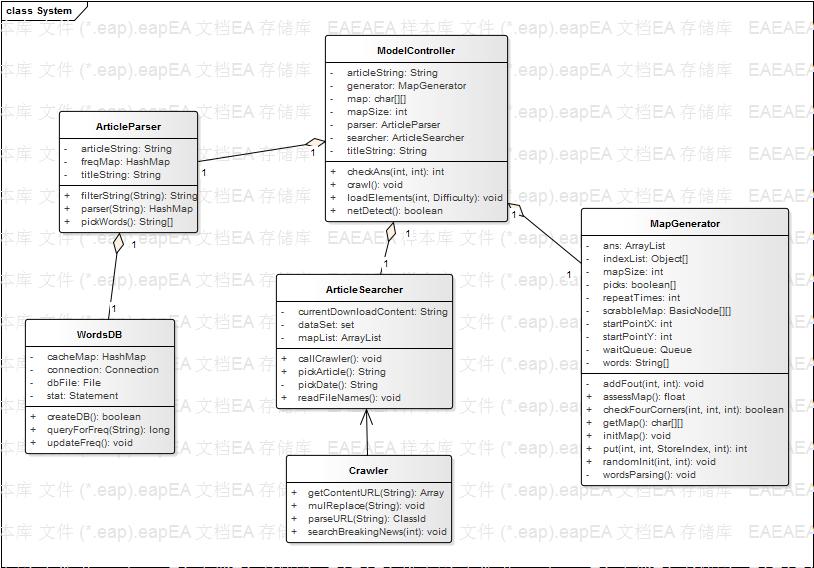
**Dyscrabble.Models.MapGenerator（地图生成器）**：其中包含对单词的索引建立、地图生成的BFS算法实现以及生成地图需要用到的子模块、地图评分及答案检测等函数。

**Dyscrabble.Models.ModelController（模块组控制器）**：包含网络检测函数以及与UI界面的数据交互模块。同时模块组控制器负责控制所有其他模型控制器的运行以及数据的交互。

**Dyscrabble.Models.WordsDB（存储控制器）**：包含对sqlite3数据库的操作函数以及Java HashMap缓存的实现操作。

**Crawler.py（网络爬虫）**：包含抓取网页所需的数据定义、文本分析与过滤、URL搜索算法实现等函数。

模块类图如下所示：



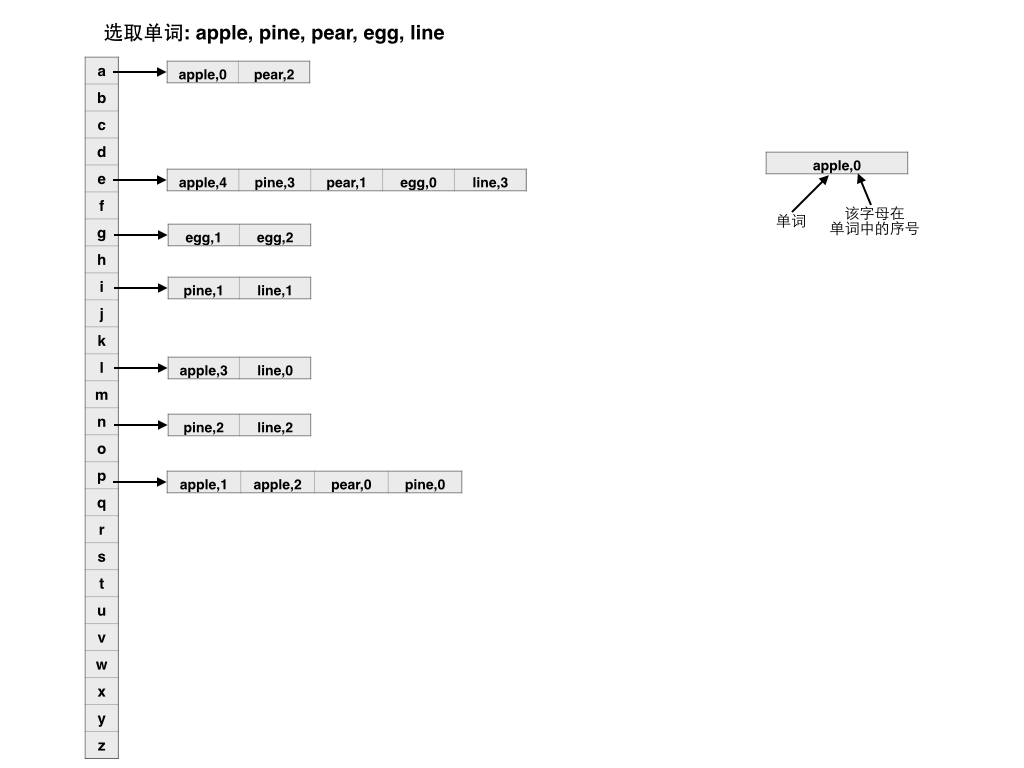
# 模块设计

在对系统的功能进行描述且对系统进行模块划分后，需要对每个模块的功能实现以及其设计思路、算法等进行详细描述。每个模块进行设计的时候，不仅仅要完成其描述的基本功能，还需要其设计的思路清晰易懂，且具有较高的效率。

## 拼字地图生成

在地图生成模块中，系统需要根据选出的单词，布置它们在地图格中的位置，使得每个单词都能与另外至少一个单词在某个字母上进行横纵交汇，即共享某个格子中的字母。该格子称为交汇点。在地图的质量方面，若能填入地图的单词数目越多（最多不超过选出的单词数），地图中横纵交汇点越多，地图的总面积越小，则质量越高。

### 字母索引表

 地图生成算法将从横纵交汇点入手。由于交汇点一个字母必须满足两个方向的单词正确性，故可以通过对每个单词的每个字母建立索引。如下图所示：

索引表中每个字母后面跟随的是拥有该字母的单词。跟随在同一个字母后面的不同单词符合在该字母处进行横纵交汇的要求。首先建立一个字母索引表。对于每个单词，分别以其每个字母作为其索引存入索引表中。

索引表中每个索引项的类定义如下：

**public** **class** StoreIndex {

**char** letter; //the indexed letter

**int** wordIndex; //index of word in the original word list

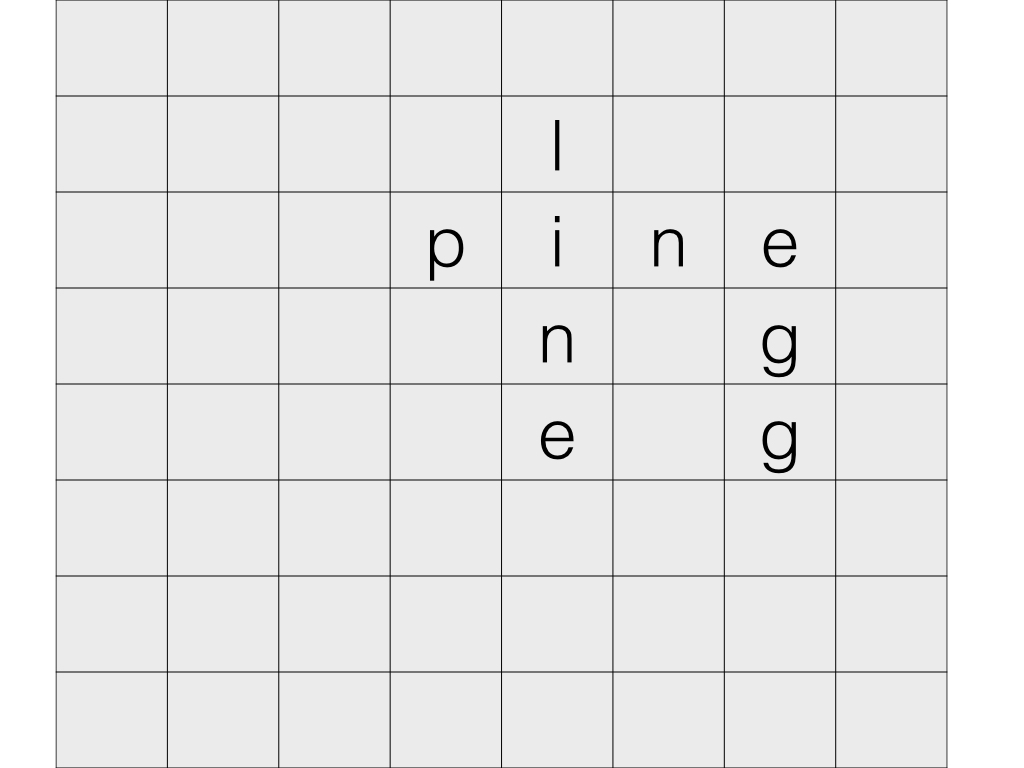
**int** alphIndex; //index of the letter in this word

**int** length; //length of this word

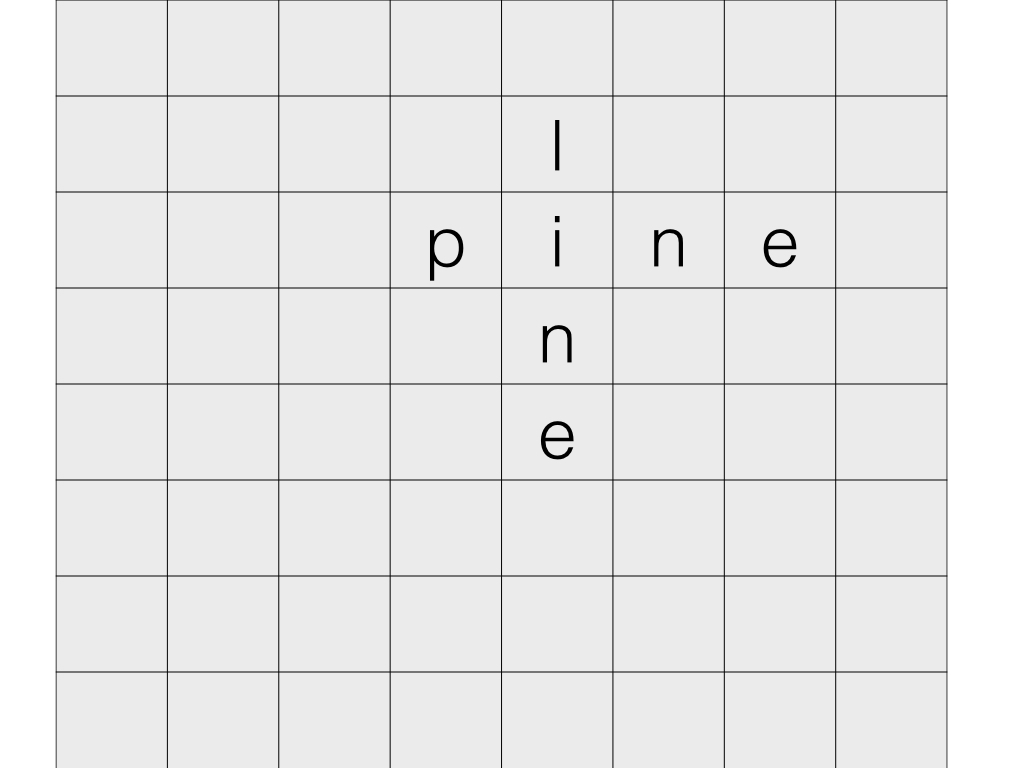
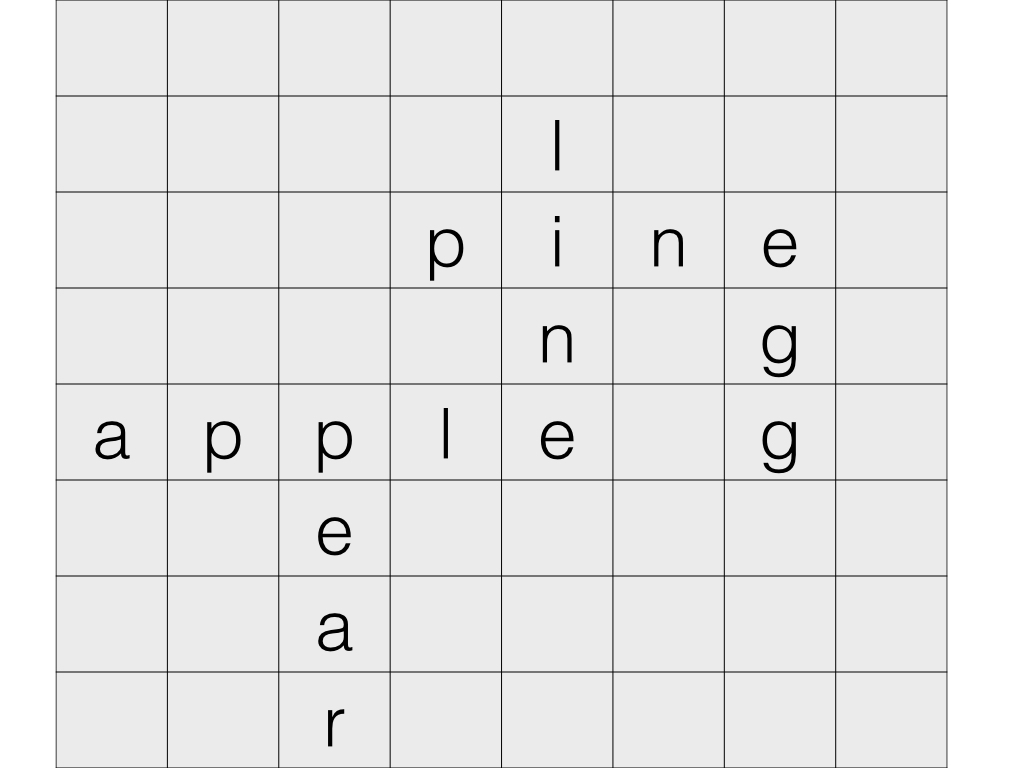
}

即每个索引项会存储该索引项的索引字母、索引单词在待填词汇表中的序号、该字母在该单词中的序号以及单词的长度。

### 地图生成算法

地图生成算法使用的是BFS算法。首先，系统将从字母索引表中随机找出至少存储了两个不同单词的字母项，如示意图中的字母i。（字母g是不符合要求的）在该字母的表项中随机选出两个单词，以字母i为中心进行横纵交叠陈列在地图的中心位置，示意图如下所示：

然后以交汇点i为起点，通过广度优先算法，将其四周的4个格子加入到队列中。每次从队列中取出头元素，查找该元素所存字母所应的索引表中是否有可以在该处填入的单词。若有，则填入后将新的交汇点也加入到队列中，直到队列为空。如下图所示：

重复使用BFS得到最终的地图：

### 填词模块

在介绍填词模块之前，需要先对地图节点的存储格式及其信息作介绍。

地图是一个的方阵。每个地图的格子称为一个节点。在图形上与一个节点共享一条边的节点称为该节点的邻节点。每个节点的类定义如下：

**public** **class** BasicNode {

**int** x; //x coordinate

**int** y; //y coordinate

**int** horizontal; //whether it is filled horizontally

**int** vertical; //whether it is filled vertically

**char** content; //the content of the node

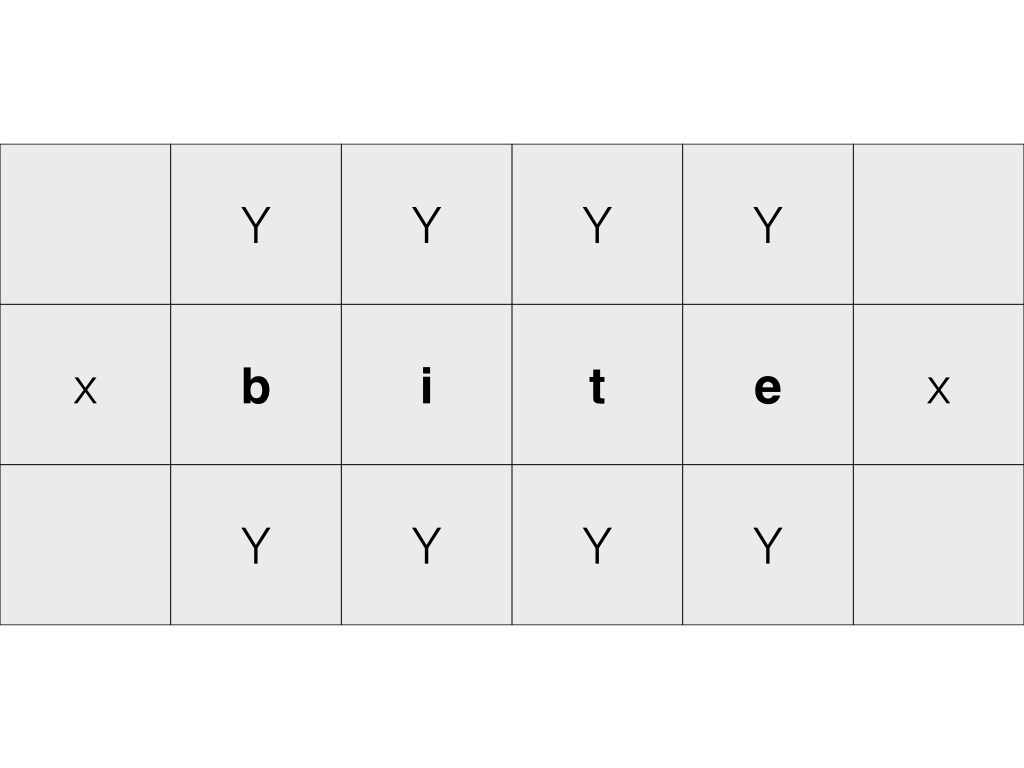
**boolean** consider; //will the node be considered by the waitQueue

}

每个节点存储该节点的横纵坐标、存储的字母、是否会被BFS考虑（后面将详细描述）、是否被横向、纵向填充（后面将详细描述）。

填词子模块主要用于尝试在某一区域填入某一个单词。造成某个单词无法填入地图主要有以下三个因素：

1. 空间不足

即给定的区域并不能将单词按字母顺序填入。有可能是待填区域中已存在字母，且该字母与待填单词填入该处的字母不一致；也有可能是由于边界限制，不能将所有字母填入。当出现空间不足时，填词失败。

2. 水平阻碍

如上图所示，待填入单词为"bite"，但是在标有“X”的区域存在已填入的字母。此时这些填入的字母会与将被填入的单词连接形成错误的单词，故在这种情况下填词失败。

3. 垂直阻碍

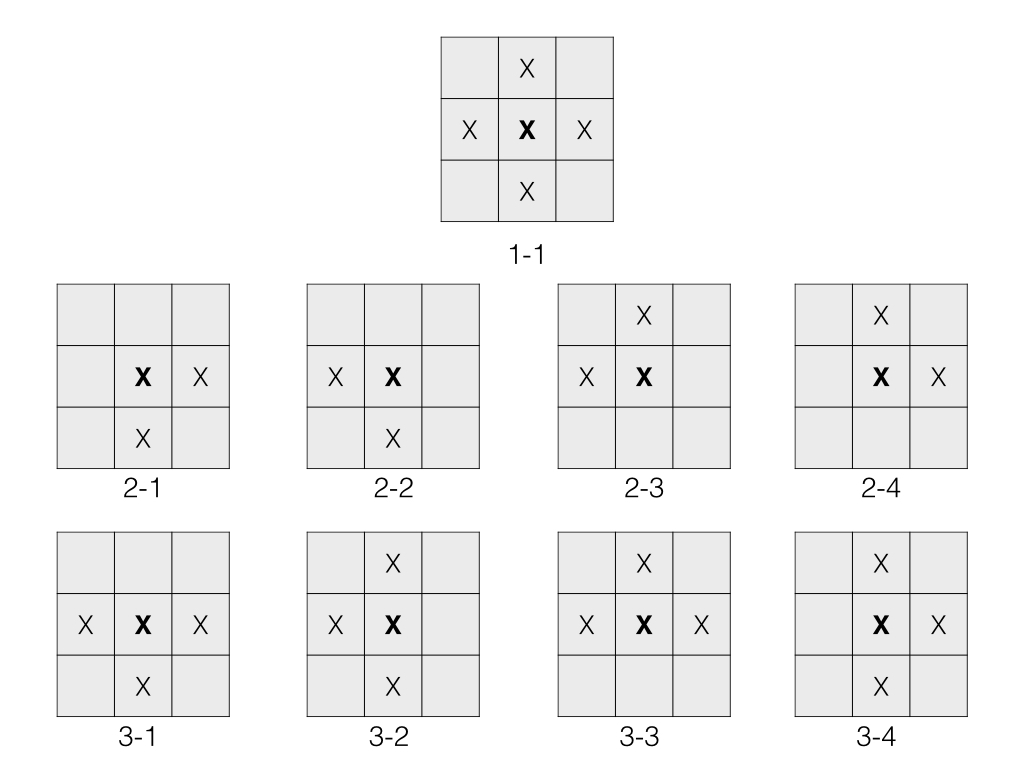
如上图所示，待填入单词为"bite"，但是在标有“Y”的区域存在已填入的字母。在不存在空间不足的情况下，这些填入的字母会与将要填入的字母在垂直方向形成许多无意义的单词。故在这种情况下填词失败。

当模块判断某个单词可以正确填入某区域时，将词填入并相应设置占据的地图节点的填入字母、横（纵）向填入情况等属性。

### 搜索邻节点模块及其优化

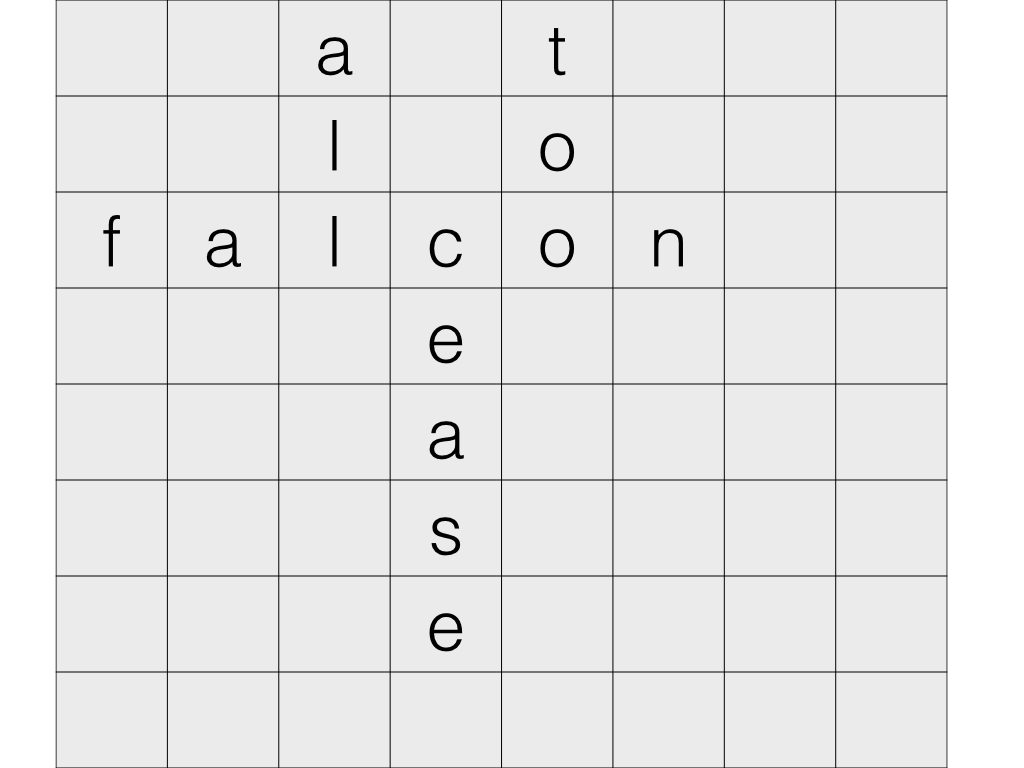
在地图生成算法中，系统会搜索当前考虑的点附近四个邻节点的情况。一般的广度优先搜索算法中，只需简单地将四个邻节点加入队列即可。大部分情况下，交汇点的四个邻节点不会再次成为交汇点，但是例外情况确实存在的；而非交汇点的四个邻节点也有可能成为交汇点。故简单地抛弃或加入四个邻节点都是不可取的，需要针对当前点附近节点的填词情况具体判断。

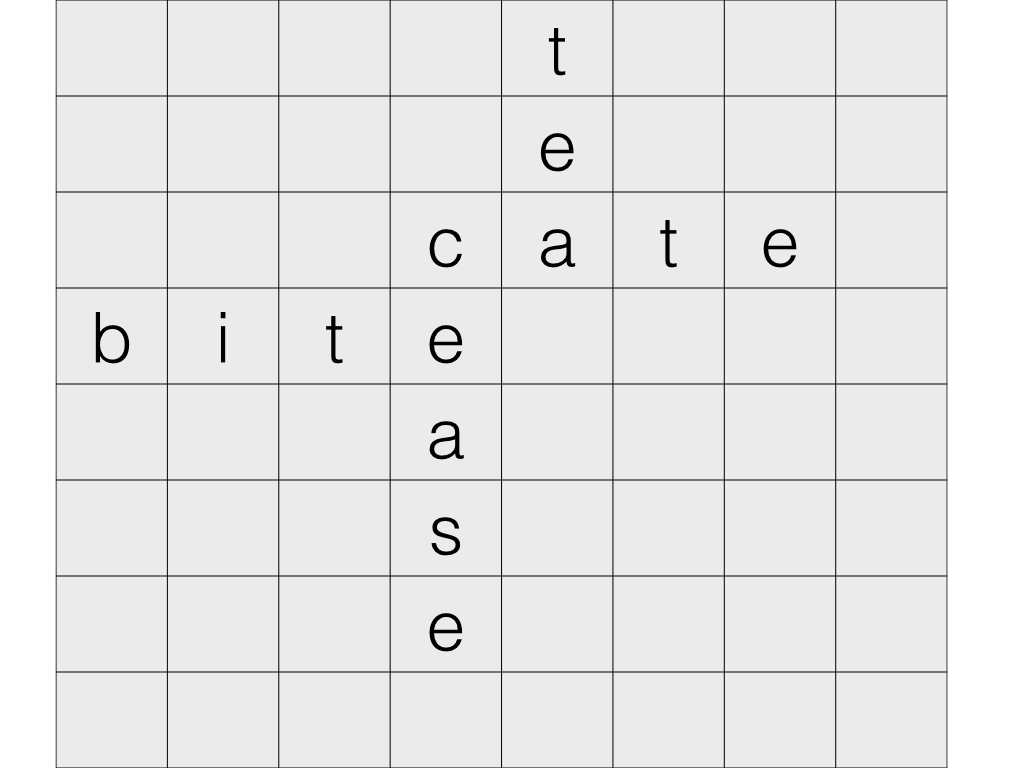
为了更好地阐述搜索邻节点模块设计思路以及其中的算法优化策略，首先对一个交汇点附近的4个节点可能出现的情况进行分类和定义。如下图所示：



图中粗体的“X”表示交汇点。可以看出，2-1、2-2、2-3、2-4是旋转对称的；3-1、3-2、3-3、3-4亦同理。由图易得，周围有2到3个字母的交汇点，必定是至少一个交汇单词的首（尾）字母。由于每个单词只有一个首（尾）字母，故这两种情况所占的比例不大。更多的情况是出现像图1-1的“十”字型。

由于系统规定选取的词汇长度不能小于3，故选取的词汇中不会出现长度为2的词汇，也即两个同方向（横向或纵向）的词汇不能粘在一起。例如，对于两个横向粘在一起（其所在行序号相差为1）的词汇来说，其纵向会产生若干个长度为2的短词，这些短词是很难组成文章中存在的词汇的。故对于一个交汇点来说，将与其距离为2的4个点加入队列中而不是距离为1的4个点能减少进入队列的无用节点数。（但是也有例外情况，将在后文进行介绍）而对于没能形成交汇点的节点，只需直接将其距离为1的四个点加入队列即可。

考虑以下两种情况：



如左图所示，若单词cate与cease的交汇点c为起点。此时与其距离为1的点e和a仍然可以作为交汇点加入单词；同理，右图中起点c的邻居点l和o也仍然能加入新的词汇。左图的情况对应交汇点图2-1到2-4；右图的情况则对应交汇点图3-1到3-4。且这些邻居点若是能够成为交汇点，也一定是新加词汇的首（尾）字母。

故每次向队列中加入新的邻节点时，首先判断交汇点情况为图中的哪一种。若是图1-1的情况（大多数情况），则直接将距离为2的点加入队列，否则，根据情况判断其邻节点是否能加入新的单词。若不能，则直接将距离为2的点加入队列。对于边界情况，可仍然视被边界截去的字母节点是存在的。在该处插入单词势必会无效，从而无需另外讨论。

这种方案能带来一定的性能提升。一般来说，“十”字型交汇点在所有的交汇点中大约占70%，对于每一个这样的交汇点，能减少4个无用的节点进入队列；对于剩下的30%情况，由于邻节点只能作为单词的首（尾）字母接入新单词，故只需判断未用的单词能否接入即可。

### 地图评分因素

上述的地图生成算法在开始阶段选取第一个交汇点的操作是随机的；节点出队列后判断能接入哪些新词也是随机的。因此每个步骤的随机化使得每一步的策略并不一定是当前最优的。（而更可能趋近于平均情况）因此需要通过多次生成的方法，筛选出相对最优的地图。

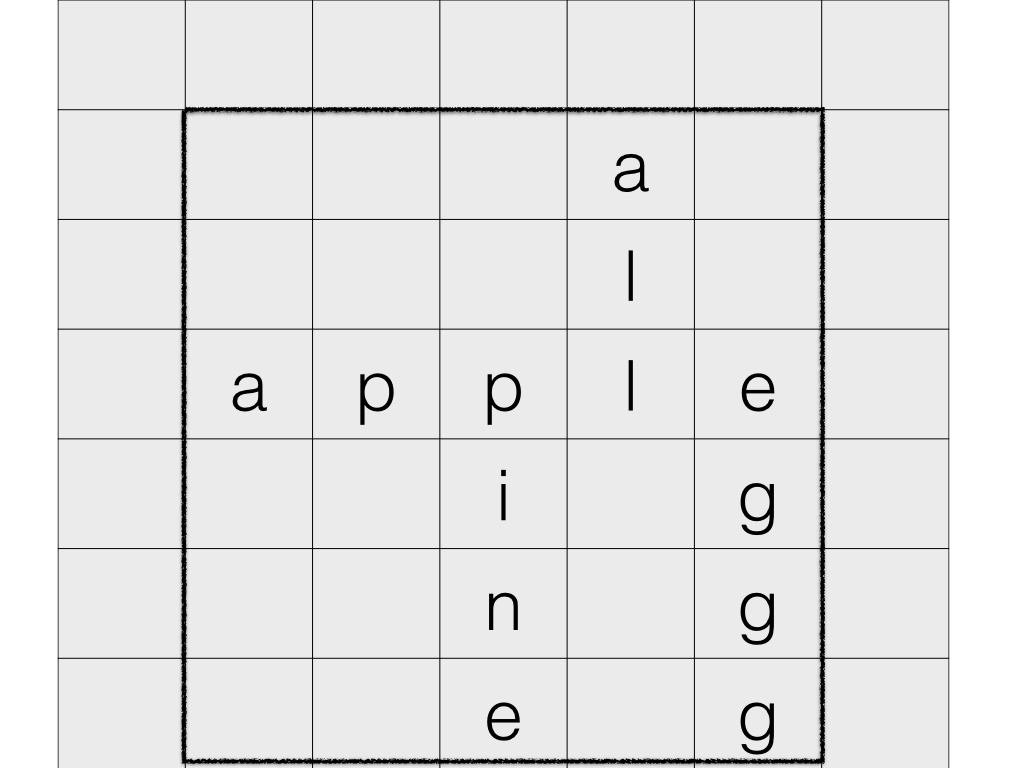
要对地图进行多次生成筛选达到优化的目的，首先需要定义衡量地图优化程度的因素。

**填入词汇数**：当队列为空时，地图生成算法便结束。故在一次生成过程中，可能由于某些步骤较为糟糕，（例如选择填入的词较长，占据了大量空间且没办法与剩下的词交汇，使得剩余的词汇不能填入到该区域）导致并不一定每个词都能填入到地图中。每填入一个单词，都会增加地图的复杂度，故填入词汇数是优化的重要因素。

**交汇点数**：生成算法的设计是的每个填入的词至少会与另外一个词交汇。即若填入n个词，则交汇点数m满足：

交汇点上的字母需要满足横纵方向单词的正确性，那么当交汇点数越多，玩家需要顾及到的单词交汇情况则越多，游戏的可玩性才越强。因此地图优化应试图最大化k值。

**地图有效矩形面积**：地图有效矩形面积即能将所有填入的单词都包含进去的最小矩形面积。如下图所示，图中粗线黑框则为地图有效矩阵。



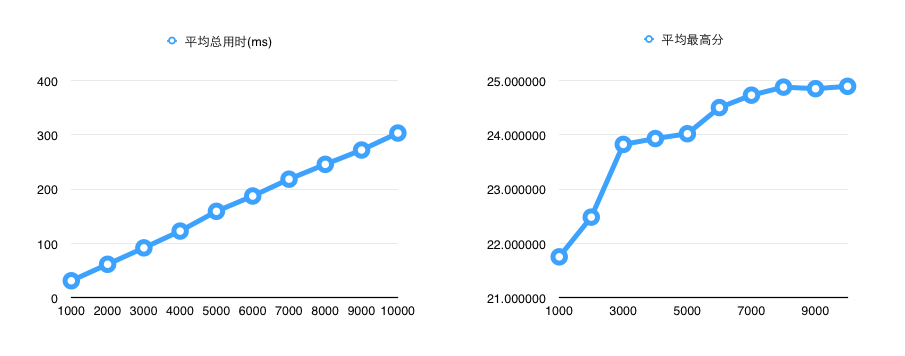
当填入的词汇数与交汇点数都一致时，若地图的有效矩形越小，表示地图越“密集”，在视觉上给予玩家的干扰效果也越好。因此地图有效矩形面积是次于交汇点数以及填入单词数的因素。

设填入单词数为n，形成的交汇点数为m，有效矩阵面积为S，拼字地图边长为w，则地图质量分数计算公式：

由上可见，单词数与交汇点的权重比为1:2，而由于必有,故，即有效矩阵因素只会在前两项均相等时才起作用。

### 地图重复生成优化策略

重复生成地图是为了选取生成的地图中质量分数最高的地图达到优化的效果。然而重复生成地图带来的时间和空间上的开销也是不可忽视的。为了选择比较适合的重复次数，我使用一篇较为典型的英语文章作为例子，（文章见附录A）统计在不同重复次数下时间的消耗以及地图质量分数上的提升。该文章为一篇GRE写作范文，其难度与从网络中下载的新闻、学术类文章难度相近，具有代表性。经过对文章单词的解析和筛选，最后选出的15个单词分别为：congruity、convivial、impressing、sketching、enmity、stressful、contradicts、aggregates、sighted、logically、divergence、temper、quarrel、disparate和contradict。下图为通过使用Java的系统函数统计的不同的算法重复次数下总用时以及最高分。（实验数据收集时重复次数皆为100次，取平均数）

由左图可知，总用时与地图生成算法重复次数基本呈现线性正相关。而当重复次数达到8000次时，最高分已基本趋于稳定。因此地图生成算法的优化重复次数设置为8000次，用时平均为245ms。

## 词汇选取与词频分析

本模块需要将文章中的每个词进行还原处理，并在词频数据库中查找并更新处理后的单词词频。之后选取词频较低的若干词汇。

词频统计即通过对词频数据库的查询找出该单词的绝对词频，在处理完毕后同时更新该词频的操作。系统中使用了轻量级的单机数据库sqlite 3进行词频的统计。在程序安装时候会从初始词频文件Lex\_ratio.txt中读取原始词频信息并创建本地数据库。在之后的游戏过程中，数据库将会根据遇到的单词相应地更新词频。

由于文章中会存在大量重复的高频常用词（如the, this），重复地对这些词汇进行数据库查询操作非常浪费时间。因此设置一个词频缓存来存储已经遇到过的单词能为查询操作提高效率。当查询一个词汇时，系统会首先查询缓存中是否已经存在该单词。若存在则直接返回词频，否则再通过数据库进行查询，并将查询的结果加入到缓存中。

查询的伪代码如下：

wordQuery(word)

if (cache.contain(word))

then

res <- cache.getValue(word) + 1

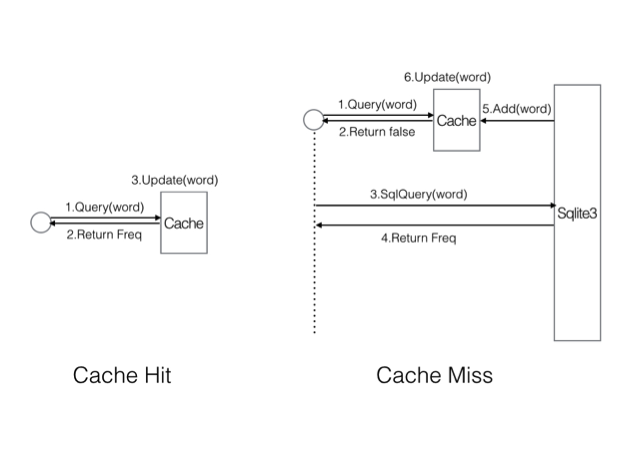
res <- cache.setValue(word)

else

res <- sqlQuery(word) + 1

cache.add(word,res)

return res

查询的示意图如下：

一篇文章的所有词汇以及其词频最终都会记录在一个HashMap中。该HashMap将用于用户填完答案后的校对检验工作。直接调用系统的sort函数并找出词频最低的若干个词汇。至此词频统计与词汇选取操作完成。

## 网络爬虫

本模块的任务为：从英文虎报官网www.thestandard.com.hk获取最近的新闻类文章并储存在目录文件中(/articles)。本模块使用Python语言编写，模块文件名为Crawler.py。该文件可直接由主程序调用与控制。从2.4中我们了解到，网络爬虫需要考虑以下三个部分：需求内容的数据定义、网页文本与URL的解析、URL搜索算法的选择。

### 需求内容的数据定义

只有明确定义了需要网络爬虫爬取的数据形式，才能定制网络爬虫的查找与解析函数并返回我们所需要的数据。在本设计中，系统需求的网页内容主要有以下几项：被下载网页的URL、下载的新闻的标题、新闻的正文内容以及新闻发布的日期。这些数据都属于文字数据，可直接在HTML文本中通过正则表达式匹配的形式一一找出。

### 网页文本与URL的解析

URL解析是URL搜索算法的前提。在thestandard官网中，网页文件的目录组织是严格按照顺序的。在HTML文档中所有该页面中存在的URL都是以"<href=URL>"的形式组织起来的。使用Python强大的正则表达式模块re能够很好地提取网页中所有的URL。但是并不是所有的URL都指向我们需要的内容。以thestandard新闻中的爆炸新闻(Breaking News)为例。thestandard中爆炸新闻的正文所在网页的URL遵循的格式为:

" http://www.thestandard.com.hk/breaking\_news\_detail.asp?id=XXXXX"

其中"id"后面为该新闻对应的id号。每篇文章都有自己的id号。所以当我们需要专门抓取爆炸新闻的URL时，只需利用正则表达式匹配上述格式的URL即可。代码如下：

pattern = re.compile(r'href="breaking\_news\_detail\.asp\?id=.+?"')

x = pattern.findall(content)

相比URL的解析，文章内容的解析显得稍微复杂些。HTML文本文件中内容繁多，想要精确定位到文章内容所在的部分，就需要找到HTML文件中正文组织的规律。所幸的是，thestandard的文件组织比较简易直观，故这种规律是存在的。查看网页的源代码后，我发现新闻的标题组织符合如下形式：

"<span class="bodyHeadline">Title</span>"

而正文则紧接在标题之后，并符合如下形式：

"<span class="bodyCopy><i>Time</i><br><br>Content</span>"

可以看出thestandard分别使用了bodyHeadline与bodyCopy样式定义了文章标题和内容的显示外观。这给网络爬虫的抓取提供了便利。由于网页还有多处使用bodyCopy样式进行定义的非正文内容，我们只能通过将标题与内容同时用一条正则表达式匹配才能分别得到正确的内容。正则表达式代码如下：

pattern = re.compile(r'<span class="bodyHeadline">(.+?)</span>.\*?<br/>.\*?<span

class="bodyCopy">.+?<i>(.+?)</i>(.+?)</span>',re.DOTALL)

contents = pattern.findall(content)

其中，第一个"(.+?)"获取的是新闻的标题，第二个获取的新闻刊登的时间，第三个是新闻的正文。

另外，由于网页编码的原因，我们需要对从网页中抓取下来的文本进行解码。首先通过Python自带的HTMLParser模块中的unescape函数获得通过iso-8859-1字符集解码后的文本，去除文本中不合法的字符，并再次编码为iso-8859-1格式储存在本地文件中。这里能防止由于网页设计不严谨导致的编码紊乱造成下载内容中出现乱码的情况。

### URL搜索算法

设计中采用BFS的URL搜索算法。BFS能保证与主页相关性较大的网页能够首先得到访问，减少无关网页的下载次数，节省带宽。使用BFS算法的时候，需要记录每一个访问过的URL。在使用urllib2模块访问当前URL之前，先将URL字符串加入到储存已访问URL的集合seen中。每次解析出新的URL时，首先判断其是否在seen集合中已出现。若出现则将其弃置。

## 文章搜寻器

本模块需要从本地文件中选取出合适的新闻文本，并将结果返回给模块组控制器。在网络爬虫模块中，本地文件名都被设定为"yyyymmdd--idxxx--breaking\_news\_detail.asp.txt"的形式，即文件名中可以展示出该新闻下载的日期、id号以及类型。搜寻器通过读取文章目录中的所有文件名，并从中选取一篇文章返回给上层模块。

在文章选取方面，搜寻器的设计采用了随机化的方法。搜寻器首先获取当天日期，并随机抽取一篇文章。若文章为当天下载的较新文章，则有85%的概率被选中；若是前一天下载的，则概率降为60%；前两天的则为25%；前三天为15%；三天以上的为5%。这种选取策略能保证选取的文章时效性仍然存在，又能使得以前的内容可以作为素材，弥补本地素材的不足。

## 其他功能模块

除了上述几大模块，系统中还有一些设计简单的小模块，用于特定的功能。

### 网络检测

网络检测用于启动网络爬虫前检测网络的状态。通过开启子进程调用ping指令对thestandard网站进行连接测试，通过返回的数据包信息判断当前网络的状况，从而确定每次网络爬虫从网站上抓取的文章数量。

### 答案检测

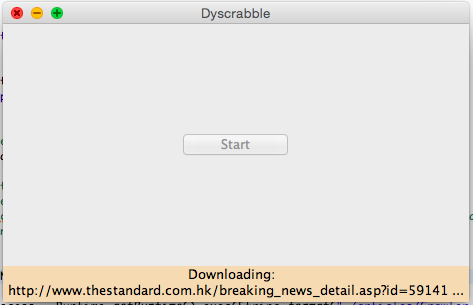
答案检测模块集成在模块组控制器中。要判断玩家输入的答案是否正确，仅仅将玩家的答案与正确答案逐一匹配是不合理的。因为若玩家在某一行（列）输入的单词A正好与标准答案在该处的单词B一样符合填字格的要求（注意A与B不是同一个词），且单词A也同时出现在了文章中，则此时应当判定玩家答案正确。要做到这一点，我们需要在ArticleParser中获取文章中的所有词汇。并以此为根据。

另一方面，通过记录每个词的开始位置、方向和长度可以定位一个单词所有字母在填字地图中的位置。如单词space的开始位置为(4,5)，方向为横向，则单词的五个字母将分别占据(4,5)(4,6)(4,7)(4,8)(4,9)。若将填入的单词都创建这种记录，那么访问所有的记录便能将地图进行遍历。

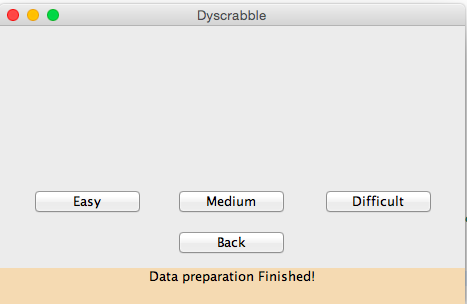
# GUI设计

在游戏设计中，GUI的设计至关重要，因为GUI将游戏的元素展示在玩家面前。游戏设计要求要有较好的交互性，力求操作简捷流畅。

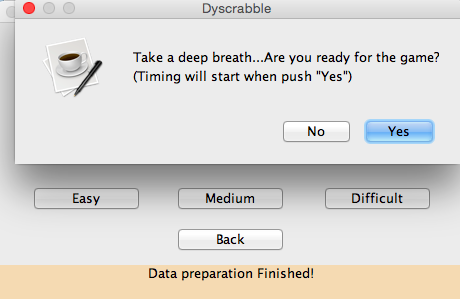
双击图标进入游戏后，游戏的开始界面如下，此时下方状态栏显示当前系统后台网络检测状况与爬虫下载的URL地址：



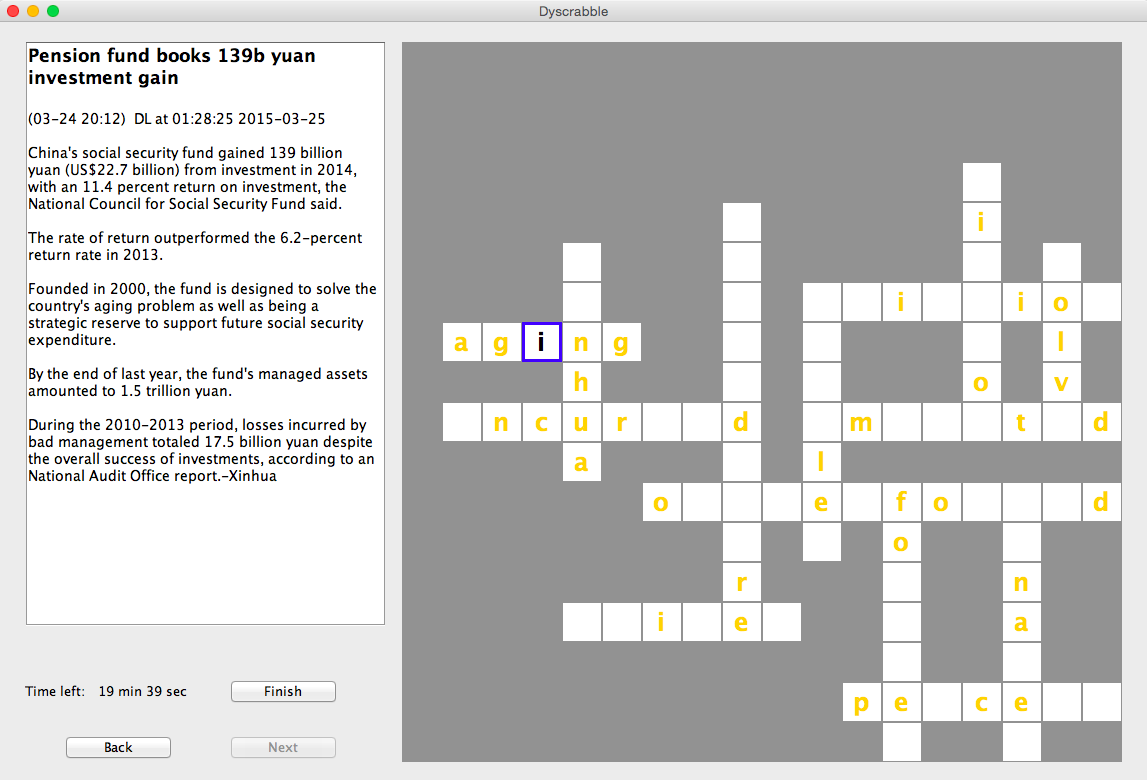
点击"Start"后进入难度选择页面：



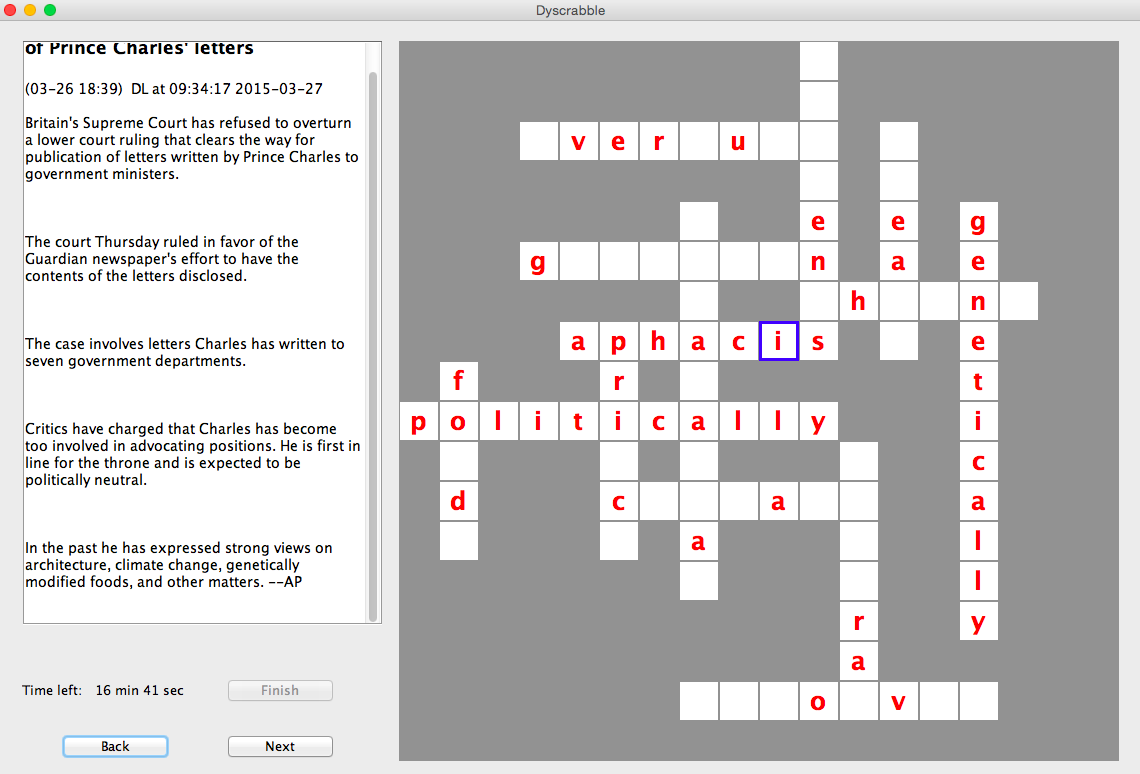
选择某一难度后，弹出对话框，需要玩家确定开始游戏：



以上开始界面只要由StartViewController进行控制。其完成的功能包括：控制开始、几个难度按钮以及返回按钮在不同的情况下的显示情况。同时，开始界面下方的状态栏需要监控后台网络检测与爬虫模块的运行情况并及时反馈给玩家。

进入游戏主界面后，左边为文章显示区。若文章较长，可拖动滚动条显示剩余部分。右边为填字地图区。当前焦点的格子用蓝色方框进行强调。玩家通过按下“上下左右”四个按钮控制焦点方框的移动，输入字母即可填入相应字母。每个格子只能填入一个字母。

点击"Finish"完成本轮游戏。此时填字地图中的字母会改变颜色。红色代表填入错误，绿色代表填入正确。



点击"Next"，进入新一轮的游戏。

# 总结与展望

本设计主要针对传统拼字游戏的玩法进行了创新，融入了实时新闻下载功能，让玩家在游戏的同时不仅能阅读到新闻内容，还能训练自己的阅读、单词联想等能力。且游戏的连贯性较传统的拼字游戏来说有所提高。设计采用了Java语言进行开发，并以Python语言编写网络爬虫脚本。设计在本地存取上使用了单机型数据库sqlite3，对英文词频信息进行了储存。并且系统在内存中设置了HashMap缓存器，增加了对数据库的访问效率。而在拼字生成算法中，系统使用了BFS算法展开地图，将单词逐个填入后生成最后的地图展示给玩家。同时设计也对BFS算法的优化进行了讨论。

在UI设计上，系统采用了Java的Swing工具包进行开发。同时将系统的模块划分为Model和ViewController两种类型。这种MVC的框架在当今中小型可视化软件开发中是十分常见的。同时，系统中采用了观察者模式、工厂模式等设计模式进行开发，更加科学地组织代码，使得代码有较好地维护性。

总的来说，本次设计涉及了多种程序语言的基础知识、数据库的访问、算法的设计与优化、UML图的设计、系统框架与设计模式等知识，是我对自己在本科所学知识的一个总结。更重要的是，在设计的过程中我也遇到了许多新的知识和难点。通过不断地上网查找资料，自己做实验等方法，这些难点得到了解决。这个过程对于我来说是十分宝贵的。

诚然，这个游戏也是有不少提升空间的。如：每一轮玩家在规定的时间内完成答案后，对于正确的部分玩家会获得分数。因此可以设计一个“英雄榜”模块，记录玩家答题正确率最高且用时最短的时刻，以此给予玩家更多的激励来进行游戏。另外，对于网络爬虫模块，由于thestandard的新闻更新速度并不快，更新的量也较少，故对于玩游戏频率较高的玩家来说，他们很容易会在短时间内碰上一篇之前读过的文章。为了避免这种情况，网络爬虫模块可以加入对其他一些英文网站的抓取功能。抓取的也可以是一些学术类文章。总之，在文章的来源和类型上进行拓展，可能会有意想不到的收获。

# 参考文献

(\* 列出你在完成论文过程主要参阅的论文与著作。其中的篇数不宜太少，否则让人感觉到你没有参考相关工作就动手做论文；一篇本科毕业论文的典型参考文献是15至30篇，如果你所列参考文献数目不足5篇，则肯定无法通过论文初审的形式检查。列举参考文献时，按论文中引用文献的先后顺序列于此处；注意正文中必须引用此处所列的全部参考文献，而且引用顺序就是参考文献的列举顺序。参考文献的著录应符合国家标准，参考文献的序号左顶格，并用数字加方括号表示，如“[1]”。每一条参考文献著录均以“.”结束。具体各类参考文献的编排格式如下：

1、文献是**期刊**时，书写格式为：

[序号] 作者. 文章题目[J]. 期刊名, 出版年份，卷号(期数):起止页码.

2、文献是**图书**时，书写格式为：

[序号] 作者. 书名[M]. 版次. 出版地：出版单位，出版年份：起止页码.

3、文献是**会议论文集**时，书写格式为：

[序号] 作者. 文章题目[A].主编.论文集名[C], 出版地：出版单位，出版年份:起止页码.

4、文献是**学位论文**时，书写格式为：

[序号] 作者. 论文题目[D].保存地：保存单位，年份.

5、文献是来自**报告**时，书写格式为：

[序号] 报告者. 报告题目[R].报告地：报告会主办单位，报告年份.

6、文献是来自**专利**时，书写格式为：

[序号] 专利所有者. 专利名称：专利国别，专利号[P].发布日期.

7、文献是来自**国际、国家标准**时，书写格式为：

[序号] 标准代号. 标准名称[S].出版地：出版单位，出版年份.

8、文献来自**报纸文章**时，书写格式为：

[序号] 作者. 文章题目[N].报纸名，出版日期（版次）.

9、文献来自**电子文献**时，书写格式为：

[序号] 作者.文献题目[电子文献及载体类型标识].电子文献的可获取地址，发表或更新日期/引用日期（可以只选择一项）.

电子参考文献建议标识：

**［DB/OL］**——联机网上数据库(database online)  
**［DB/MT］**——磁带数据库(database on magnetic tape)  
**［M/CD］** ——光盘图书(monograph on CD-ROM)  
**［CP/DK］**——磁盘软件(computer program on disk)  
**［J/OL］** ——网上期刊(serial online)  
**［EB/OL］**——网上电子公告(electronic bulletin board online)

\*)

1. 袁崇义. Petri网原理. 北京：电子工业出版社. 1998.
2. Bloch J.. Effective Java: Programming Language Guide. Addison Wesley. 2001.
3. 李建中, 李金宝, 石胜飞. 传感器网络及其数据管理的概念、问题与进展. 软件学报. 2003, 14(10): 1717-1727.
4. Goguen J. A.. Parameterized Programming. IEEE Transactions on Software Engineering. 1984, 10(5): 528-543.
5. Snyder A.. Encapsulation and Inheritance in Object-Oriented Programming Languages. In Proceedings of Annual ACM SIGPLAN Conferences on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA’86). 1986. 38-45.
6. … …

# 

# 致 谢

(\*谢辞应以简短的文字对课题研究与论文撰写过程中曾直接给予帮助的人员(例如指导教师、答疑教师及其他人员)表示对自己的谢意，这不仅是一种礼貌，也是对他人劳动的尊重，是治学者应当遵循的学术规范。内容限一页。\*)

# 附 录

(\*对于一些不宜放在正文中的重要支撑材料，可编入毕业论文的附录中。包括某些重要的原始数据、详细数学推导、程序全文及其说明、复杂的图表、设计图纸等一系列需要补充提供的说明材料。如果毕业设计(论文)中引用的实例、数据资料，实验结果等符号较多时，为了节约篇幅，便于读者查阅，可以编写一个符号说明，注明符号代表的意义。附录的篇幅不宜太多，一般不超过正文。

论文附录依次用大写字母“附录A、附录B、附录C……”表示，附录内的分级序号可采用“附A1、附A1.1、附A1.1.1”等表示，图、表、公式均依此类推为“图A1、表A1、式A1”等。

\*)