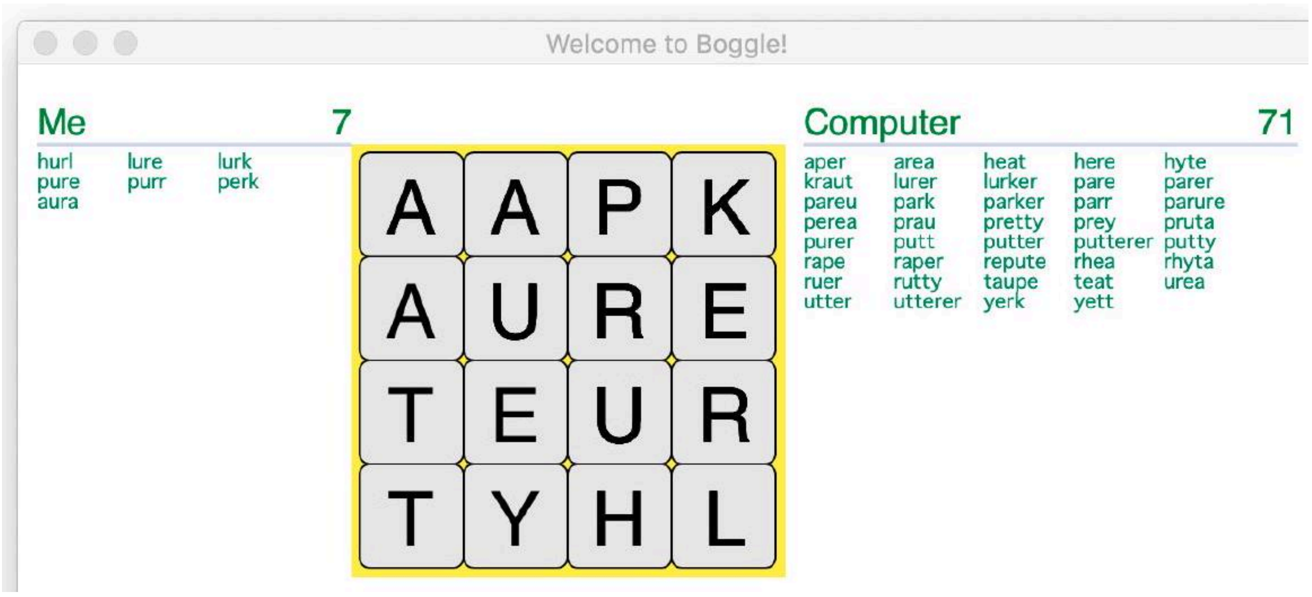
Assignment 3: Boggle 作业 3：博格智力游戏

Thanks to Todd Feldman for the original assignment idea. And thanks to Julie Zelenski and Eric Roberts for the handout.

感谢托德-费尔德曼（Todd Feldman）提出最初的作业构思。感谢 Julie Zelenski 和 Eric Roberts 提供的讲义。



```
Me                 7
hurl   lure   lurk
pure   purr   perk
aura

Computer                                      71
aper    area     heat    here      hyte
kraut   lurer    lurker  pare      parer
pareu   park     parker  parr      parure
perea   prau     pretty  prey      pruta
purer   putt     putter  putterer  putty
rape    raper    repute  rhea      rhyta
ruer    rutty    taupe   teat      urea
utter   utterer  yerk    yett
```

The Game of Boggle 博格游戏

Those of you fortunate enough to have spent summers seeing the world from the back of the family station wagon with the ‘rents and sibs may be familiar with Boggle, the little word game that travels so well, and those who didn’t will soon become acquainted with this vocabulary-building favorite. The Boggle board is a 4×4 or 5×5 grid over which you shake and randomly distribute 16 or 25 dice. These 6 -sided dice have characters instead of numbers, creating a grid of letters from which you can form words. In the original version, the players start simultaneously and write down all the words they can find by tracing by a path through adjoining letters. Two letters adjoin if they are next to each other horizontally, vertically, or diagonally. There are up to eight letters adjoining a cube. A letter can only be used once in the word. When time is called, duplicates are removed from the players’ lists and the players receive points for their remaining words based on the word lengths.

那些有幸在夏天和父母及兄弟姐妹们一起坐在家用旅行车后座上看世界的人，可能对 Boggle 这个非常适合旅行的小文字游戏并不陌生。Boggle 游戏板是一个 4×4 或 5×5 网格，在网格上摇动并随机分配 16 或 25 个骰子。这些 6 面骰子上有字符而不是数字，形成了一个字母网格，您可以用这些字母组成单词。在最初的版本中，玩家同时开始游戏，通过在相邻字母间划出一条轨迹，写下他们能找到的所有单词。如果两个字母在水平、垂直或对角线上相邻，它们就会相邻。一个立方体中最多有八个字母相邻。一个字母在单词中只能使用一次。时间到后，重复的字母将从玩家的列表中删除，玩家将根据剩余单词的长度获得相应的分数。

Due: Wednesday, October 16th at 5:00 pm

提交时间：10 月 16th 日星期三下午 5:00

Assignment 3 asks that you write a program that plays a fun, graphical rendition of this little charmer, adapted to allow the human and machine to play one another. As you can see from the screen shot above, the computer generally pummels you into the ground, but it’s fun to play anyway, and the assignment frames it as an interesting recursion problem. The main focus of the assignment is designing and implementing the recursive algorithm required to find all the words that appear in the Boggle board.

作业 3 要求你编写一个程序，以有趣的图形方式演绎这个小游戏，并允许人机对弈。从上面的屏幕截图中可以看出，电脑一般都会把你打得落花流水，但不管怎样，玩起来还是很有趣的，而作业也把它归结为一个有趣的递归问题。作业的主要重点是设计和实现所需的递归算法，以找出 Boggle 棋盘中出现的所有单词。

How's this going to work?

这怎么行得通?

You set up the letter cubes, shake them up, and lay them out on the board. The application precomputes all words that can be formed and pairs them with information about where they appear. The human player gets to go first (nothing like trying to give yourself the advantage). The player enters words one by one. After verifying that a word is legitimate, you highlight the word's letters, add it to the player's word list, and award the player some points.

你可以摆好字母方块，摇一摇，然后把它们放在黑板上。应用程序会预先计算出所有可组成的单词，并将它们与出现地点的信息配对。人类玩家先下手为强（没有什么比努力为自己争取优势更重要了）。玩家逐个输入单词。在确认一个单词是合法的之后，您会高亮显示该单词的字母，将其添加到玩家的单词列表中，并给予玩家一些分数。

Once the player has found as many words as they can, the computer gets its turn. The computer identifies (and takes credit for finding) all the remaining words and awards itself all points. The computer typically beats the player mercilessly, but the player is free to try again and again and again until he or she feels inferior. ()

当玩家找到尽可能多的单词后，就轮到电脑了。电脑会找出（并记下找到的）所有剩下的单词，并为自己赢得所有分数。电脑通常会毫不留情地击败玩家，但玩家可以自由地一次又一次地尝试，直到感到自卑为止。()

The letter cubes 字母方块

The letters in Boggle are not simply chosen at random. Instead, the letter cubes are designed in such a way that common letters come up more often and it is easier to get a good mix of vowels and consonants. To recreate this, our starter code declares an array of the cubes from the original Boggle. Each cube is described using a string of 6 letters, as shown below:

Boggle 中的字母并不是随意选择的。相反，字母方块的设计使常见字母出现的频率更高，也更容易获得元音和辅音的良好组合。为了重现这一点，我们的启动代码声明了一个由原始 Boggle 游戏中的方块组成的数组。如下所示，每个立方体都用一串 6 个字母来描述：

```
static const string kStandardCubes[16] = {
    "AAEEGN", "ABBJOO", "ACHOPS", "AFFKPS",
    "AOOTTW", "CIMOTU", "DEILRX", "DELRVY",
    "DISTTY", "EEGHNW", "EEINSU", "EHRTVW",
    "EIOSST", "ELRTTY", "HIMNQU", "HLNNRZ"
};
static const string kBigBoggleCubes[25] = {
    "AAAFRS", "AAEEEE", "AAFIRS", "ADENNN", "AEEEEM",
    "AEEGMU", "AEGMNN", "AFIRSY", "BJKQXZ", "CCNSTW",
    "CEIILT", "CEILPT", "CEIPST", "DDLNOR", "DDHNOT",
    "DHHLOR", "DHLNOR", "EIIITT", "EMOTTT", "ENSSSU",
    "FIPRSY", "GORRVW", "HIPRRY", "NOOTUW", "OOOTTU"
};
```

These strings are used to initialize the cubes on the board. At the beginning of each game, “shake” the board cubes. There are two different random aspects to consider. First, the cubes themselves need to be shuffled so that the same cube is not always in the same location on the board. Second, a random side from each cube needs to be chosen to be the letter facing up.

这些字符串用于初始化棋盘上的方块。每局游戏开始时，都要 "摇动 "棋盘上的方块。有两个不同的随机方面需要考虑。首先，需要对立方体本身进行洗牌，这样同一个立方体就不会总是出现在棋盘上的同一个位置。其次，需要从每个立方体中随机选择一面作为朝上的字母。

To rearrange the cubes on the board, you should use the following shuffling algorithm, presented here in pseudo-code form:

要重新排列棋盘上的方块，你应该使用下面的洗牌算法，这里以伪代码的形式呈现：

```
Copy the constant array into a vector vec so you can modify it.
Shuffle vec using the following approach:
    for (int i = 0; i < vec.size(); i++) {
        Choose a random index r between i and the last element position, inclusi
        Swap the element at positions i and r
    }
Fill the Boggle grid by choosing the elements of vec in order.
```

This code makes sure that the cubes are randomly distributed across the grid. Choosing a random side to put face-up is straightforward. Put these two together and you can shake the cubes into many different combinations.

这段代码可以确保方块在网格中随机分布。随机选择一面朝上也很简单。把这两部分放在一起，你就能把方块摇成多种不同的组合。

Alternatively, the user can choose to enter a custom board configuration. In this case, you still use your same board data structure. The only difference is where the letters come from. The user enters a string of characters, representing the cubes from left to right, top to bottom. Verify the string is purely alphabetic and of a length capable of filling the board, and re-prompt if it is too short.

用户也可以选择输入自定义的电路板配置。在这种情况下，您仍然可以使用相同的电路板数据结构。唯一不同的是字母的来源。用户输入一串字符，从左到右、从上到下代表方块。验证字符串是否纯粹是字母，长度是否能够填满棋盘，如果太短，则重新提示。

Precomputing all formable words

预计算所有可造词

Once the board has been populated with letters, you should recursively search the board to find all words. A word is considered legitimate if:

在棋盘上填入字母后，应递归搜索棋盘，找出所有单词。在下列情况下，一个单词被认为是合法的

It is at least four letters long.

至少有四个字母长。

It is contained in the English lexicon.

它包含在英语词典中。

It can be formed on the board (i.e., it is composed of adjoining letters and each cube is used at most once).

它可以在棋盘上组成（即由相邻字母组成，每个立方体最多使用一次）。

When a word can be formed in multiple ways, you can just ignore any possibilities beyond the first, Because a word can only be counted once during play, it's only important you present some path on the board when the word is player. Words that differ only in capitalization scheme are considered the same (e.g. "tree", "TREE", and "TrEe" are all the same word).

当一个单词可以有多种组成方式时，可以忽略第一种之外的任何可能性。由于一个单词在游戏过程中只能被计算一次，因此只有当该单词是玩家时，您才有必要在棋盘上显示一些路径。仅大小写不同的单词被视为相同单词（例如，"tree"、"TREE "和 "TrEe "都是相同的单词）。

As with any search algorithm with a high branch factor, it's important to find ways to limit the search to ensure that the process can be completed in a reasonable amount of time. One of the most important Boggle strategies is to prune dead end searches. For example, if you have a path starting **zx**, the Lexicon's containsPrefix method should inform you there are no English words down that path. So, you should stop right there and move on to more promising search paths. If you ignore this suggestion, you'll find yourself taking long coffee breaks while the computer is looking for words like zxgub, zxaep, etc.

与任何具有高分支因子的搜索算法一样，重要的是要找到限制搜索的方法，以确保在合理的时间内完成搜索过程。最重要的 Boggle 策略之一就是剪除死胡同搜索。例如，如果您有一条以 **zx** 开始的路径，词典的 containsPrefix 方法就会告诉您这条路径上没有英文单词。因此，你应该就此打住，继续寻找更有希望的搜索路径。如果你忽视了这一建议，你就会发现自己在长时间喝咖啡的同时，电脑还在寻找 zxgub、zxaep 等单词。

The human's turn 轮到人类

Once everything's been precomputed, you allow the user to enter as many words as they can find. When they enter something bogus, your program should reject their response with a helpful message saying why it was rejected (too short, not an English word, not on the board, been used before, etc.) using whatever message you want to use.

一旦一切都预先计算好了，你就可以让用户尽可能多地输入他们能找到的单词。如果用户输入的是假词，程序就会拒绝他们的回答，并给出一条有用的信息，说明拒绝的原因（太短、不是英语单词、不在黑板上、以前用过等等）。

If all is good, you add the word to the player's word list and score. In addition, you graphically show the word's path by temporarily highlighting the relevant cubes. You can use the graphics function pause to implement the delay. The length of the word determines the score: one point for a 4-letter word, two points for 5 letters, and so on. The functions from the gboggle.h interface provide helpful routines for highlighting cubes, displaying word lists, and handling scores.

如果一切顺利，则将该单词添加到玩家的单词列表中并得分。此外，您还可以通过暂时高亮相关方块，以图形方式显示单词的路径。您可以使用图形函数暂停来实现延迟。单词的长度决定得分：4 个字母的单词得 1 分，5 个字母的得 2 分，以此类推。gboggle.h 界面中的函数为高亮显示方块、显示单词列表和处理分数提供了有用的例程。

The player enters a blank line when done finding words, which signals the end of the human’s turn.

找完单词后，玩家输入一行空行，表示人类回合结束。

The computer's turn 轮到计算机

The computer then searches the entire board to find the words not found by the human. The computer’s already done all the precomputation, so it can take credit for all the words it precompiled and claim them as its own. The only words it can’t post as its own finds are those the human player found during his or her turn.

然后，计算机会搜索整个棋盘，找出人类没有找到的单词。计算机已经完成了所有的预计算，因此它可以将所有预编译的单词归功于自己，并声称这些单词是它自己找到的。只有人类玩家在自己的回合中找到的单词，才不能作为自己找到的单词发布。

Our provided code 我们提供的代码

We have written all the fancy graphics functions for you. The functions exported by the gboggle.h interface are used to manage the appearance of the game window. It includes functions for initializing the display, labeling the cubes with letters, highlighting cubes, and displaying the word lists. The implementation is provided to you in source form so you can extend this code in your own novel ways if you are so inclined.

我们为您编写了所有花哨的图形函数。gboggle.h 接口导出的函数用于管理游戏窗口的外观。它包括初始化显示、用字母标记方块、高亮显示方块和显示单词表的函数。我们将以源代码的形式为您提供实现方法，您可以根据自己的喜好以新颖的方式扩展这些代码。

The Grid and Lexicon classes you’ve already seen will come in handy again. Our English “dictionary.txt” file contains over 250,000 words, which is about six times as large as the average person’s vocabulary, so it’s no wonder the computer is so good.

你已经见过的网格和词典类将再次派上用场。我们的英语 "dictionary.txt "文件包含超过 25 万个单词，大约是普通人词汇量的六倍，难怪电脑这么好用。