Computing A Level Project: French Crossword Game

Frederic Higham
Bolton School Boys' Division

Wednesday 8th November, 2023

Contents

1 Analysis

1.1 Overview

The aim of this project is to create an effective program that generates French crosswords for my French teacher, Ms Roddy, to use in teaching. The game should be used to help her students revise vocabulary and possibly grammar through challenging them to solve crosswords made up of a large range French words pertaining to the A-level curriculum.

1.2 The Problem and Why it is Being Solved

Learning new languages is very difficult for school students. Having to learn the vocabulary, grammar, spellings and pronunciations can be overwhelming. Many will agree that simply staring at textbooks is not a great way to revise grammar and vocabulary. Ms Roddy, indicated that her students often have trouble remembering the vocabulary they have learned and could do with a resource to help[1]. To help solve this issue, I proposed making a French learning game, which I intend will improve upon other learning games in some regards.

Studies have shown that serious games[2] as well as gameification[3] are greatly beneficial for language learning [5]. This is because video games are interactive; they take input from the user and respond to it in real time, possibly using AI to adapt the experience (like how *Quizlet* [9] favors testing you on the vocabulary it knows you are struggling with). Video games are also very accessible and can be used practically anywhere (making them a great tool for virtual learning; something which is becoming increasingly common with COVID-19 [1]). In one of the interviews, my teacher indicated that, in her experience, games can be quite effective in language teaching. Because of these factors, I believe the creation of my own French learning game will be very useful for my teacher. As well, being an A level French student myself, this game should be of use to my studies.

1.3 Process of Finding a Solution

Initial inspirations were taken from websites and apps like *Languages Online* [8], *Quizlet* [9] and Duolingo [3]. These all feature a range of different games and features designed to help users practice and learn languages. I wanted to create a program that had multiple different game modes the users could play from and which tested different skills such as grammar and vocabulary. My French teacher and some of her students

[1] I interviewed Ms Roddy in person on several occasions, one time including some Y11s in the conversation. The conversations were recorded and the transcripts can be found in the "Transcripts" section.

[2] Serious games are video games created specially for the purpose of education [4].

[3] Gameification is the process of adding game mechanics into non-game environments [6].

suggested the game be able to help with listening, be able to track progress and to be linked to the curriculum. Dealing with all these aspects in one program would be beyond the scope of this project, so after careful consideration, I decided to focus the program on one game mode for practicing one skill: a crossword game for practicing vocabulary. My French teacher supported this, saying that crosswords would be great tool for vocab learning.

The crosswords found at *Languages Online* [8] (a website with multiple activities designed to help people learn languages) served as an inspiration for this project. I identified multiple issues with the site's crosswords and decided to try making an improved version. These issues include: the limited range of vocabulary difficulty options, a limited number of crosswords (it does not randomly generate new ones each time), little user control (the user can not decide the size of the grid etc.) and an unintuitive interface. I decided to try to make an improved version of this. To do so, I have looked at different online crossword sites such as the *Guardian* [2] to see how the crosswords are laid out, what kind of clues they use (such as "1 across is another word for ...") and how the interfaces are designed.

1.4 Description of Chosen Solution

Many of the crosswords I found at the *Guardian* feature very complex grid patterns and clues which I recognise would be very difficult to code. Therefore, I will greatly limit the complexity of the crossword grids (the grid size and number of words) and the clues (it will simply ask you to translate words). Figure 1 shows an example of a grid my crossword generation algorithm should be able to fill in. To further limit complexity, the program will not have to generate crossword grid patterns from scratch (except as an extension task). Instead, the program will select a random grid pattern from a file of pre-made grids which it will then try to fill in with words. So, looking at Figure 1 for example, the program would try to find a five letter word to fit into "1 across" and then an eight letter long word to fit into "2 down" which has an intersecting letter with "1 across". Once the algorithm has filled in the grid pattern with appropriate words, the user can then try to fill in a blank copy. The program will ignore accents as is the custom with French crosswords (it would be too difficult to construct crosswords which were accent sensitive). Words with spaces in them will be joined together in the crossword with no space.

A GUI menu system will need to be designed which can allow the user to select the specifications (topic, crossword language, grid size, ...) they wish to play with for the crossword. Once the program has generated a crossword, it will need to be able to represent it graphically on the screen alongside all of its clues. Every word line in the grid should be allocated a number that should displayed in the top corner of the first gird square in the line (see the "word line numbers" in Figure 1). Each square on the crossword grid should be able to be selected by the player's mouse and then be typed into with a letter. Once the player has filled in the crossword there should be a button they click which checks if they are right and, if not, tells them where they went wrong.

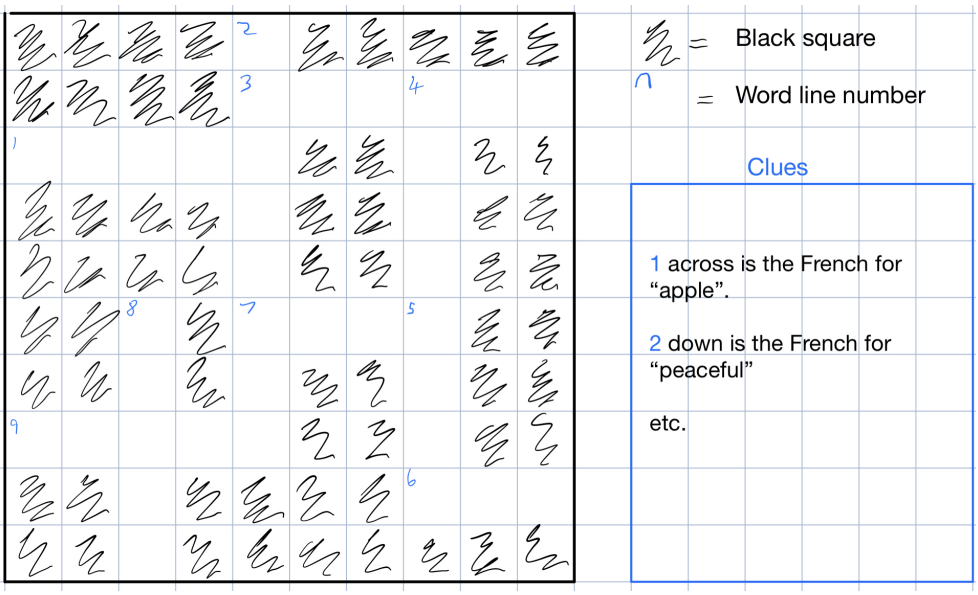


Figure 1: An example of a 10x10 crossword grid the program would fill in and the clues it would produce.

In addition, another button should be available that the player can click if they are stuck which will give them some help (such as filling in some of the missing letters in the crossword).

A database will need to be created which contains all of the French words being used. These words should cover a range of A level topics and have varied lengths and letters. This because crosswords look better when they have varied word lengths and if there were only two 7 letter words then the pattern matching would be much more difficult. Multiple tables will need to be used which contain amongst them the following attributes: English translation, topic, word class (noun, adjective, ...) and gender (so that it knows whether to use "un" or "une".). The aim is that whatever specifications the user chooses, the program will fetch appropriate words (at random) from the database to fill the crossword in with.

My first extension task will be to generate more complex crossword clues, something my teacher thought would be a good step forward. This will mean storing in the database extra attributes such as synonyms of the words and extra details to be used for clues. For the second extension task, I will try to generate blank crossword patterns from scratch rather than using pre-made ones. It will need to be able to generate ones which match the size the user inputs and which allow a range of word lengths. The final extension task will be to factor in the user's progress into the game, another feature my teacher requested. This means the program would record which words the user is struggling with and which ones they have correctly filled in. And using this information, the program could be more likely to use the words the user has struggled on and less likely to use the ones they have correctly entered for all of the subsequent

crosswords.

1.5 Infrastructure required

Python will be used as the programming language for this project because it is simple but effective and I have a lot of experience using it. The *Pygame* graphics package [7] will be utilised because it is appropriate for the project's scope and easy to use. In addition, the Sqlite3 module [10] will be utilised to construct SQL queries for my database of French words. The game will be able to be run on any machine which runs Mac, Windows or Linux operating systems and which has both Python and the Pygame module installed. There should not be any difficulty in my teacher or her students being able to play this game at school because the necessary infrastructure (computers, Python, . . .) is in place and they are all experienced with using computers.

1.6 List of Objectives

The following is a numbered list of primary and extension objectives for the algorithm:

1. Store data.
    - 1.1 Have a file filled with pre-made blank crossword grids for the program to use.
        - 1.1.1 There should be a range of differently arranged and sized crossword grids.
    - 1.2 Have a database made up of all the French words being used.
        - 1.2.1 Be as efficient and accurate as possible.
        - 1.2.2 Contain all the necessary attributes (English translation, topic, word class and gender).
2. Create a menu system.
    - 2.1 The user needs to be able to select the following options in the menu:
        - 2.1.1 Grid size.
        - 2.1.2 Crossword language.
        - 2.1.3 Topic.
    - 2.2 They need to be able to return to the menu at any time with a button.
    - 2.3 They need to be able to quit the program at any time.
3. Fill in the crosswords.
    - 3.1 Needs to fetch a suitable crossword grid from the file of blank crossword grids.

3.2 Needs to fill the crossword in with suitable words (right length for each line, right topic, ...).

3.2.1 Should fetch these words from the database.

3.2.2 Each word should only be used once.

3.3 Allocate each word line a number.

4. Present the crossword for the user.

4.1 Display a blank version of the crossword on the screen for the user to fill in.

4.2 Display the word line numbers in the top corners of the first gird squares on the line.

4.3 Display all of the clues on the side as well as the word line numbers they correspond to.

4.4 Each square on the crossword grid should be able to be selected by the player's mouse and then be typed into with a letter.

4.4.1 The user should only be able to type in appropriate characters (no symbols, numbers or accents).

4.5 Provide a button for checking if the user's answers are correct.

4.6 Provide a button to ask for help.

5. *Generate more complex crossword clues (extension task).*

5.1 *Rather than simple translations, use clues such as "2 across is the synonym of 'pomme'" or "2 across is a red object you can eat".*

5.2 *Will require more attributes to be stored in the database such as synonyms and descriptions of the nouns (colour, what you do with it, ...).*

6. *Generate crossword patterns from scratch (extension task).*

6.1 *Based on the user's specifications, generate a random pattern to fill in (rather than using a pre-made one).*

7. *Factor in the user's progress (extension task).*

7.1 *Create a list of all the words the user is struggling on and all of the ones they have correctly filled in.*

7.2 *Be more likely to use the words they have struggled on and less likely to use the ones they have correctly filled in for the subsequent crosswords.*

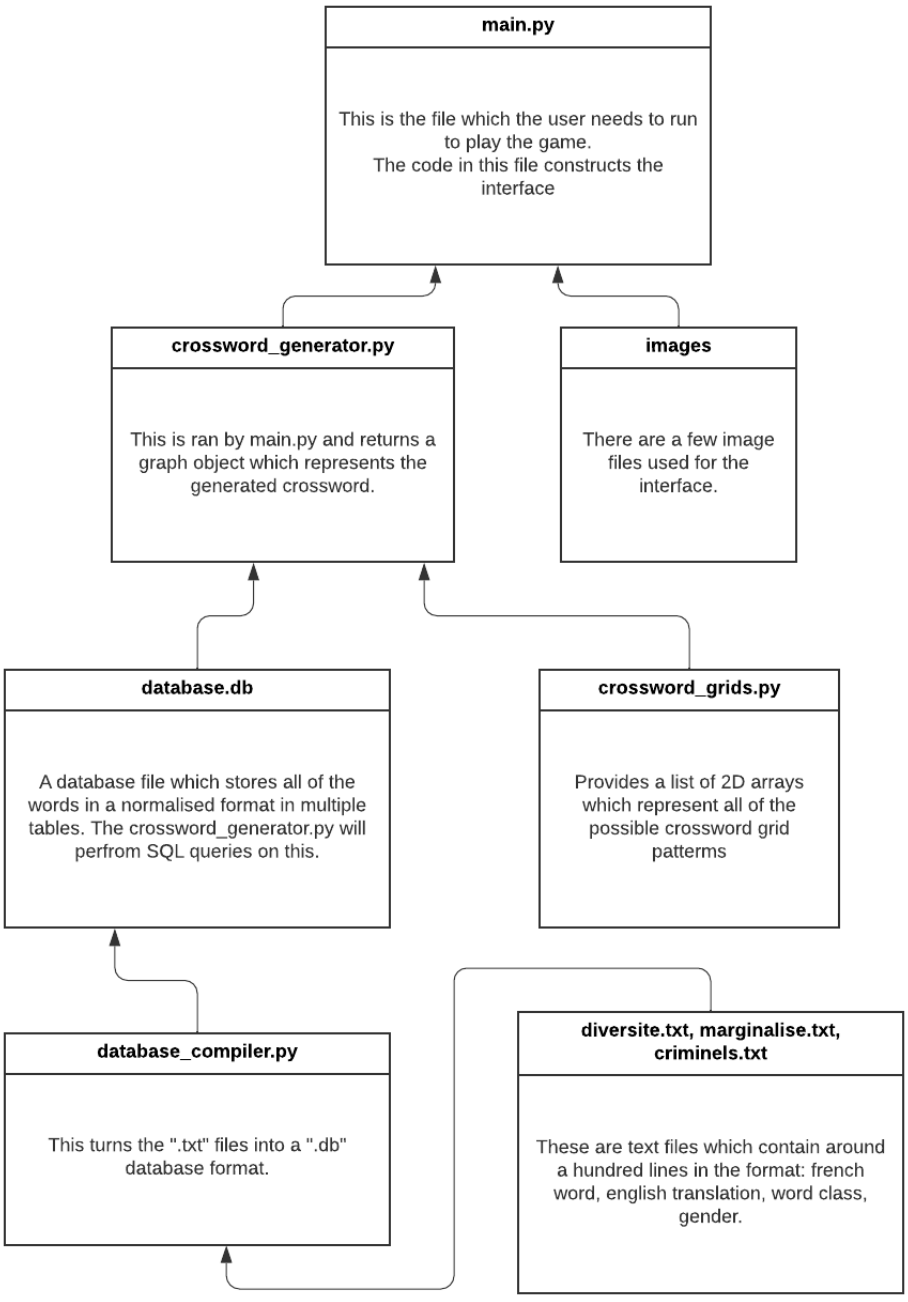


Figure 2: A diagram which outlines the different files used for the project and their purposes. The arrows indicate which files are used by which .

2 Documented Design

2.1 Overview of Approach

Figure 2 gives an overview of the different files which will be used for this project and what their purposes will be. The following sections describe in more details the approach for each major aspect of the design.

2.2 Database

A database will be constructed to store all of the A level French words which will be used to form the crosswords. It will be made using the Python module *Sqlite3*, a module which allows you to construct SQL queries using Python, and it will be stored in a '.db' file. The words will all be taken from the AQA French A level textbook.

2.2.1 Tables and Relationships

The program will use the following four normalised tables:

- FrenchWords (FrenchWordID, FrenchWord, FrenchWordForCrossword, WordClass, Topic, WordLength)
- FrenchGender (FrenchWordID, Gender)
- EnglishWords (EnglishWordID, EnglishWord, EnglishWordForCrossword, WordLength)
- Translations (FrenchWordID, EnglishWordID)

Figure 3 shows the relationships between these normalised tables. Each word can have multiple translations but any translation can only involve one French word and one English word. Therefore a one to many relationship has been drawn. Similarly, each French word can have only one gender therefore a one to one relationship has been drawn. The atributes FrenchWordID and EnglishWordID are both primary keys. They will be formed by using numbers in order beginning from 0 (so the first word will have the key 0 then the next will be allocated 1). "EnglishWordForCrossword" and "FrenchWordForCrossword" are versions of the words with characters removed to make them more appropriate for the crossword. These characters include spaces, accents, articles such as "le" and "la" and the "to" used for English infinitives. The following SQL code will be used to construct the tables:

Example Code 2.1: SQL CREATE TABLE examples.

```
1 CREATE TABLE FrenchWords(
2 FrenchWordID INTEGER NOT NULL PRIMARY KEY,
3 FrenchWord VARCHAR(20),
4 FrenchWordForCrossword VARCHAR(20),
5 WordClass VARCHAR(10),
```

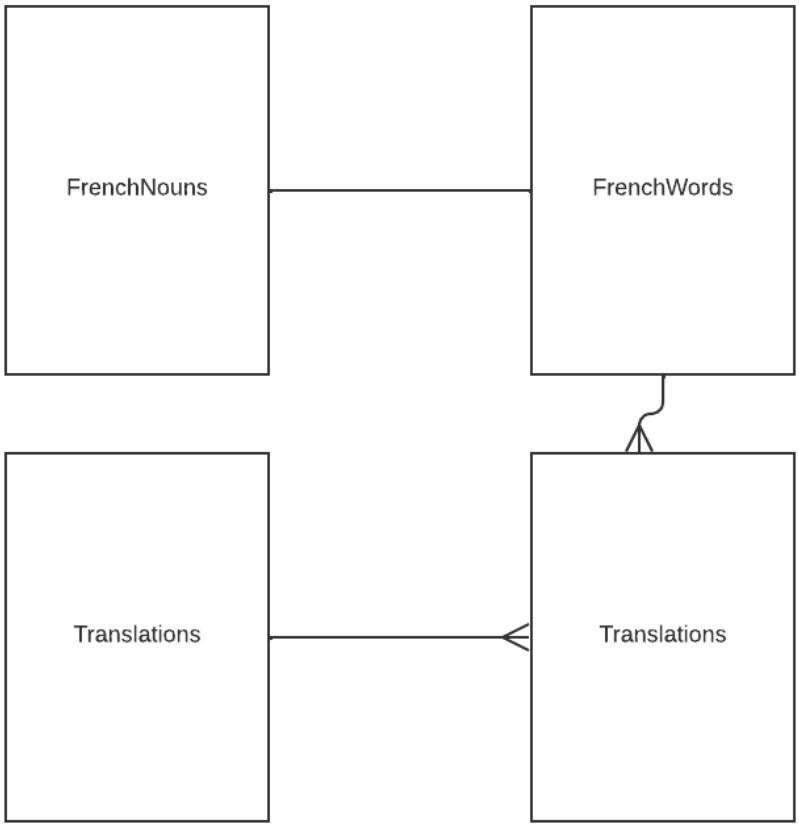


Figure 3: A relationship diagram for the database.

```
6   Topic VARCHAR(20),
7   WordLength INTEGER)
8
9   CREATE TABLE EnglishWords(
10  EnglishWordID INTEGER NOT NULL PRIMARY KEY,
11  EnglishWord VARCHAR(20),
12  EnglishWordForCrossword VARCHAR(20),
13  WordLength INTEGER)
14
15  CREATE TABLE Translations(
16  FrenchWordID,
17  EnglishWordID,
18  FOREIGN KEY (FrenchWordID) REFERENCES FrenchWords(FrenchWordID),
19  FOREIGN KEY (EnglishWordID) REFERENCES EnglishWords(EnglishWordID))
20
21  CREATE TABLE FrenchGender(
22  FrenchWordID,
23  FOREIGN KEY (FrenchWordID) REFERENCES FrenchWords(FrenchWordID),
24  Gender VARCHAR(20))
```

2.2.2 Creating The Database

An efficient way needs to be found for filling the database in with all of the French words being used. Thousands of records in 4 different tables will need to be created each with multiple attributes. I initially came up with the idea of creating a program that prompts me to enter the data record by record (using the Python "input()" function) running an appropriate SQL command with each record I input to add to the database. The benefit was that this program would avoid me having to keep writing SQL commands to fill in the database and it would automatically generate the FrenchWordForCrossword (by removing the spaces and accents in the words I input). The problem was that this still took too long because I had to manually type in each attribute for each word from the textbook and if I wanted to edit a word I would have to manually construct SQL UPDATE commands to do so.

Instead I have decided to use a different approach where I fill a ".txt" file with all the input data and then "compile" this using a program. Each line in the file will contain the attributes for one record in the format: FrenchWord, EnglishWord, WordClass, Gender (optional). Section 3.5 shows a file which the program would "compile". By "compile" I mean that the program will read through the file line by line and construct an appropriate SQL UPDATE query for each. This is better than the previous method as it is far easier to type all of the words into a text file than it is to do so through Python "input()" prompts. Furthermore, it is more convenient to edit the database as all I need to do is edit the text file then re-run the program.

2.2.3 Queries

The crossword generating algorithm will need to construct SQL queries to fetch the specific words that fit into each word line. So, looking at Figure 4, if the program wants to find a word for 2 down, a query will need to be made which searches for a French word of the correct topic, with a length 8 letters long and which has the letter "e" as its third letter (because it intersects with the word "pomme"). Here is the SQL code for this query:

Example Code 2.2: Query example.

```
SELECT FrenchWords.FrenchWord, EnglishWords.EnglishWord,
FrenchWords.WordClass
FROM FrenchWords, EnglishWords, Translations
WHERE (FrenchWords.FrenchWordID = Translations.FrenchWordID)
AND (EnglishWords.EnglishWordID = Translations.EnglishWordID)
AND (FrenchWords.WordLength = 8) AND
(SUBSTR(FrenchWordForCrossword, 3,1) = 'e')
AND (FrenchWords.topic='fruit')
ORDER BY RAND () LIMIT 1;
```

Figure 4: An example of a crossword. It has partially been filled in.

2.3 Crossword Generating Algorithm

2.3.1 Crossword Grids

The blank crosswords templates that the program will use to fill in will be stored in a separate file. It is important that there is a range of different grid sizes and arrangements. They will be represented using a 2 dimensional array like so (where "X" represents black squares and spaces represent letter spaces):

Example Code 2.3: 2D crossword grid.

```
crossword_pattern_1 =
[["X","X",X,X","  ","X","X","X","X","X"],
["X","X","X","X","  ","  ","  ","  ","  ","  ","  ","  "],
["  ","  ","  ","  ","  ","  ","  ","X","X","  ","  ","X","X"],
["X","X","X","X","  ","  ","X","X","  ","  ","X","X","X"],
["X","X","X","X","  ","  ","X","X","  ","  ","X","X","X"],
["X","X","  ","X","  ","  ","  ","  ","  ","  ","X","X","X"],
["X","X","  ","X","  ","  ","X","X","  ","  ","X","X","X"],
["  ","  ","  ","  ","  ","  ","X","X","  ","  ","X","X","X"],
["X","X","  ","X","X","X","X","X","  ","  ","  ","  ","  "],
["X","X","  ","X","X","X","X","X","X","X","X","X","X"]]
```

A dictionary data structure will also be employed to organise the crosswords by size as so:

Example Code 2.4: Crossword grid dictionary.

```
grids = {"10x10": crossword_pattern_1,
         "11x11":crossword_pattern_2, ...}
```

2.3.2 Graph Traversal Approach

The crossword generating algorithm will begin by choosing a random crossword grid to fill in based on the grid size specified by the user. Before the crossword can be filled in, it needs to be converted from a 2D array to a graph form, an example of which can be seen in Figures 5 and 6. In order to do this it will loop though the 2D array of the blank crossword grid and map out where the word lines are and at which positions they intersect. Each node on the graph will represent one of these word lines and store attributes such as its direction (across or down), start/end position and word line number. The edges on the graph will represent which word lines intersect with each other and where they do so.

Once the graph has been constructed, the program will pick which node to start depth-first traversing from (probably beginning with the smallest word line number i.e. "node 1"). For this node, an appropriate SQL query will be produced which can find a list of words which suit the space it represents on the crossword grid (needs to be a word of the correct length, language and topic as specified by the user). One of the words from the query result will be chosen to work with and the rest will be stored for later use. If no suitable words can be found for the first node then the traversal ends and a new crossword grid is tried. If at least one word has been found, the program will then traverse to another node which the first node shares an edge with and perform another SQL query. This time the query will need to include letter specifications. For example, if the first node has the word "apple" and the second node intersects with it in the second position then it will need to find a word that begins with "p". If it finds at least one result then it will traverse to the next node and repeat the process. If no results are ever found, then the previous node visited (indicated by a visited stack) will be returned to and a different word from its query result will be tried. Once every word from the query result of a previous node has been tried then it will traverse even further backwards to the node before that and keep trying new words.

Recursion will be used to loop through the traversal as it works well with depth-first traversal and, in my opinion, makes for clearer code. There will be a traversal method which takes as the parameters the current node and the visited stack and will be repeatedly run each time the program wants to traverse to another node. The base case of the traversal is either if every word from the query result of the first node has been tried or if the crossword has been successfully filled in. Figure 5, shows a pseudocode representation of this traversal algorithm.

Example Code 2.5: main recursive graph traversal algorithm.

```
1
2  PRIVATE PROCEDURE traversal(current-node, visited)
3
4      needed-letters ← calculateNeededLetters(current-node)
5      traverse ← False
6
7      IF current-node.doesNotHaveSavedQuery() THEN
8          result ← performQuery(needed-letters, current-node)
9          IF result == [] THEN
10             IF visited != [] THEN
```

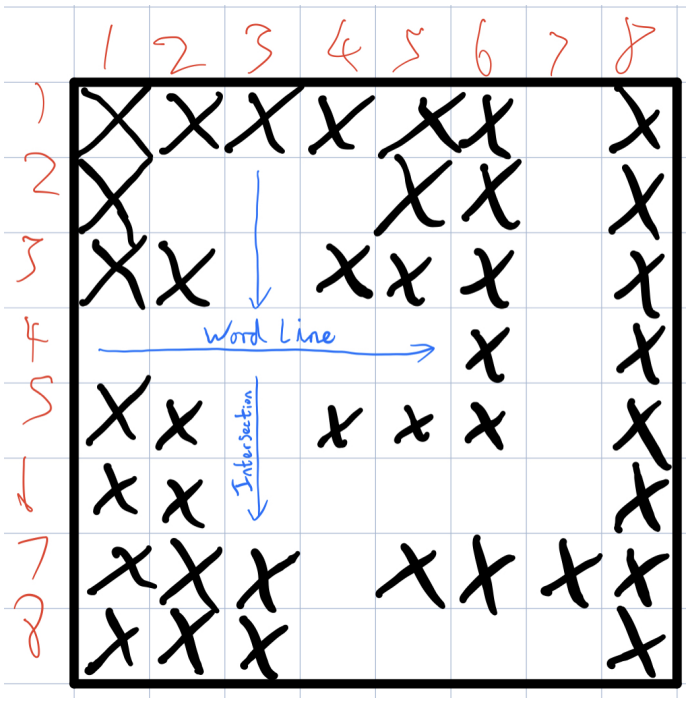


Figure 5: An example of a crossword grid used to form the graph in Figure 6

```
                visited-store ← visited[-1]
                visited.dequeue()
                traversal(visited-store, visited)
            ENDIF
        ELSE
            current-node.saveQueryResult()
            traverse ← True
        ENDIF

    ELSIF NOT current-node.doesNotHaveSavedQuery() and
    current-node.everySavedQueryResultTried() THEN
        IF visited != [] THEN
            current-node.resetNode()
            visited-store = visited[-1]
            visited.dequeue()
            traversal(visited-store, visited)
        ENDIF

    ELSE
        current-node.SwitchToNextQueryWord()
        traverse ← True
    ENDIF

    IF traverse THEN
        visited.append(current-node)
        FOR intersection IN current-node.getIntersections()
            IF intersection[0] NOT IN visited:
                traversal(intersection[0], visited)
```

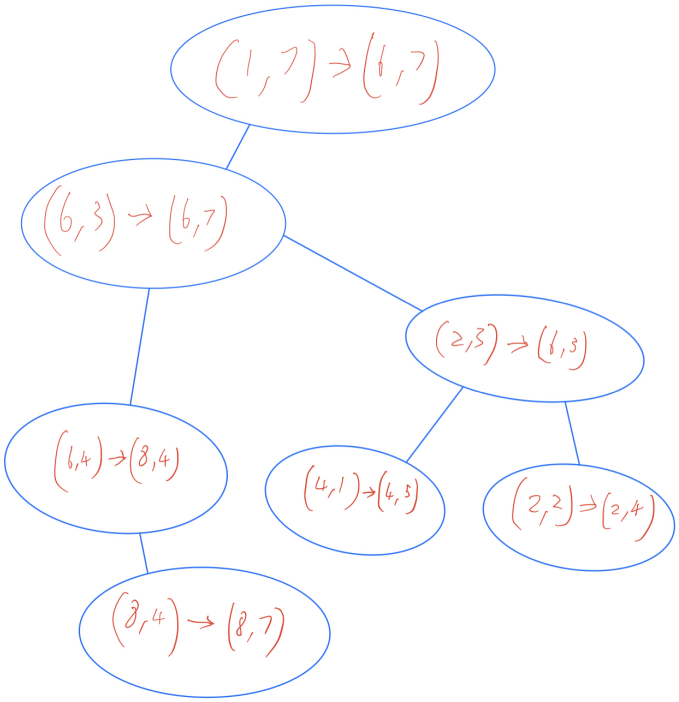


Figure 6: An example of a graph (based on the grid in Figure 5) which the program would produce (the coordinates are in the form (row, column)).

2.3.3 OOP

I have decided to use OOP to represent the graph and nodes because it will result in better structured and easier to debug code. Figure 7 shows a diagram containing the two objects used for this algorithm (*WordLine* and *Graph*) as well as their attributes and methods. The *Graph* class will deal with representing and constructing the crossword grid and will be composed of *WordLine* objects. Each *WordLine* object will represent a word on the crossword grid and can be thought of as nodes in the graph. The classes will be encapsulated, meaning that all of the attributes will be private and will only be able to be interacted with by using public get and set methods.

2.3.4 Limitations

There are various limitations with my chosen approach:

1. The limited number of words in the database means there are only a few "solutions" for filling in any given crossword grid.

2. The program must try every permutation of words until it finds one that fits the crossword grid; it doesn't prioritise trying certain kinds of words which are more likely to work, therefore it is inefficient.

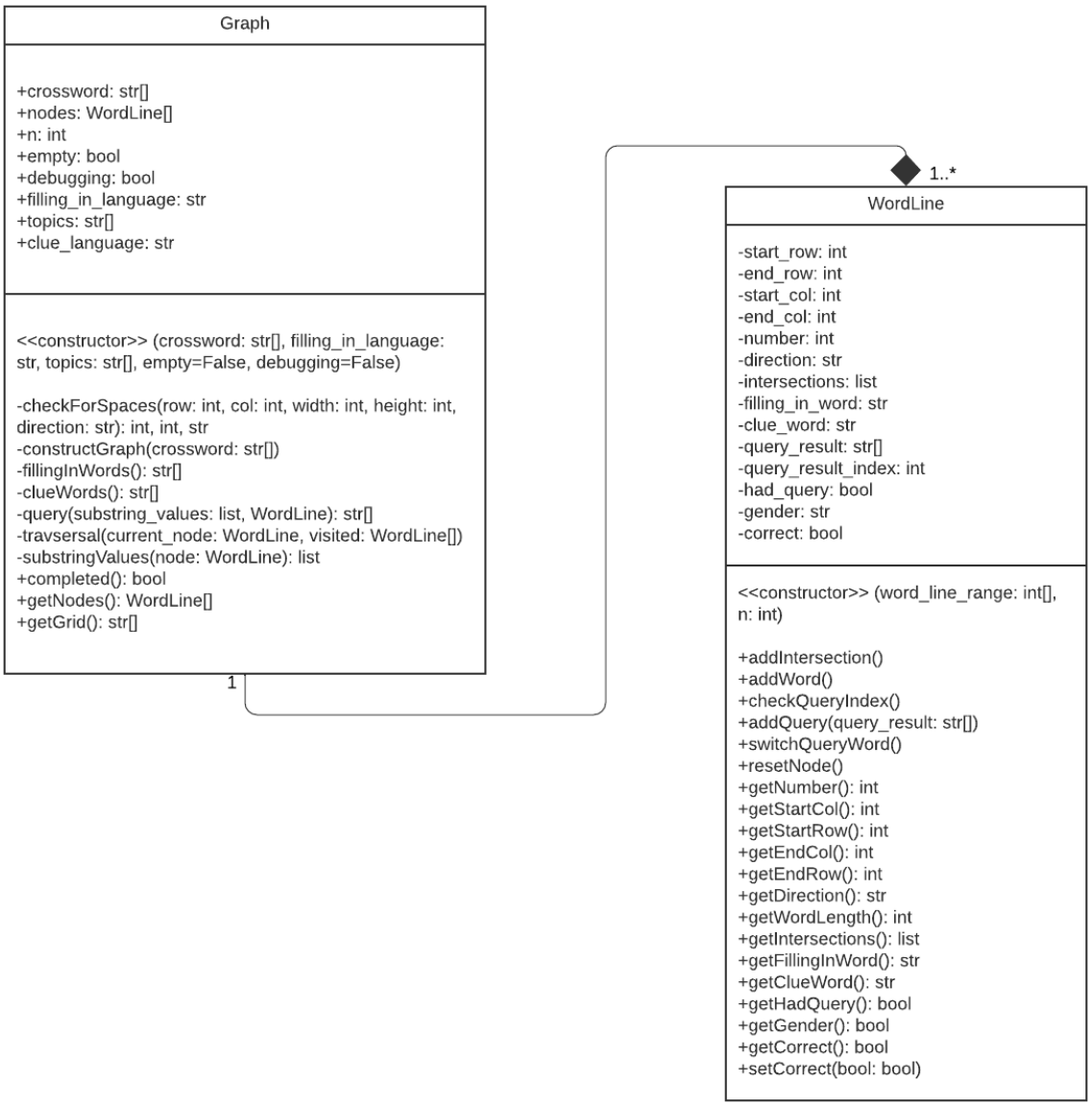


Figure 7: A UML class definition diagram for the crossword generating algorithm.

3. The use of recursion means there is the threat of reaching the call stack limit for large and complex crossword grids.

An alternative I explored was for the algorithm to dynamically generate its own crossword patterns as it tries to fill it in (rather than attempting to fill in a static grid). This would be more effective as it could create crossword patterns which suited the words rather than the other way around (thus solving problem 1). I could also prioritise certain words over others which have vowels in certain places which make them easier to connect with other words.

I have decided to stick with the chosen approach because, after testing it, I found that it worked despite being theoretically inefficient; there was no need to spend further time developing this alternative approach (which would be far more complex than the chosen approach). In order to combat the call stack limit error, I could use exception handling to catch the error and abort the process. I could also limit the complexity of the crossword grids such that the algorithm is more likely to find a solution.

2.4 GUI Program

The GUI will be split into two screens: the menu screen and the crossword screen. The program will keep looping between the two screens and if one screen is closed, the other will open. I know the interface will be an important aspect in creating an effective learning game so I will need to spend a lot of time making it as interactive and user friendly as possible.

2.4.1 Menu Screen

The aim of this screen is to allow the user to specify what kind of crossword they want to be generated for them. There will be three sets of toggle buttons to decide what crossword size, topic and filling in language they want. Once the user has selected their preferences they should be able to press a button which will take them to the crossword screen where they can see their generated crossword. Figure 8 shows a draft of what the screen should look like.

2.4.2 Crossword Screen

In this screen there will be an interactive crossword grid on which the user can select any grid square with their mouse and enter in a letter. They will also be able to use the arrow keys to change which grid square is selected. When they think they are finished they can press the check crossword button (this can only be pressed once) and a score will be displayed (the number of correct words entered by the user out of the total number of words on the crossword). Any grid square that contains an incorrect letter will have the correct letter displayed in the colour red to indicate to the user where they went wrong.

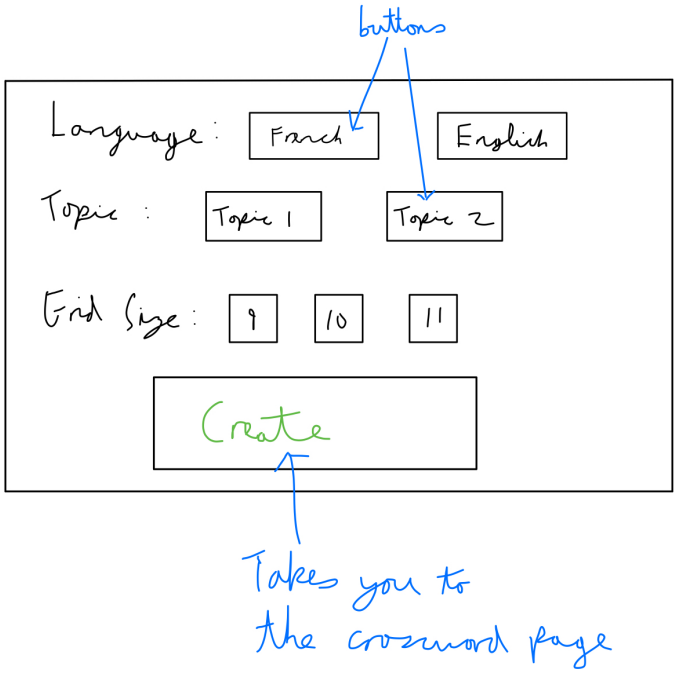


Figure 8: A design for the menu screen.

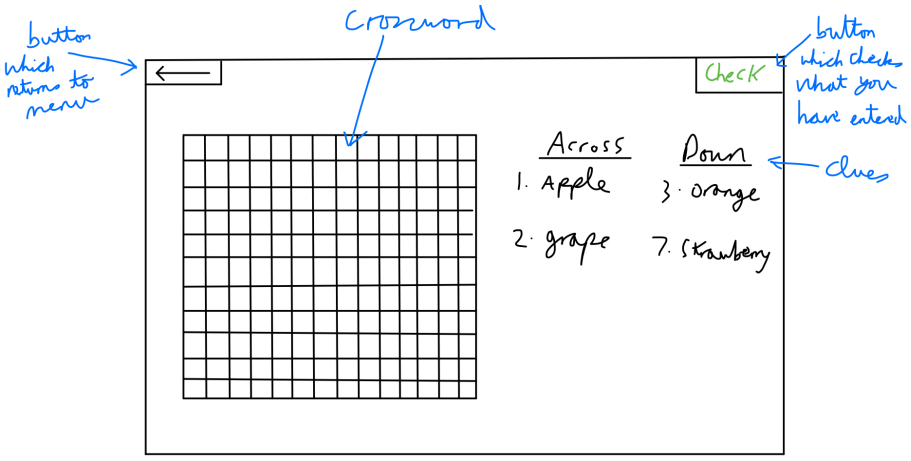


Figure 9: A design for the crossword screen.

2.4.3 Implementation in OOP

The GUI program will be completely coded in the object oriented paradigm for the same reasons mentioned in section 2.3.3. Figure 10 shows a UML class definition diagram for this program. Both the menu screen and crossword screen will be classes composed of buttons, text and grid square objects.

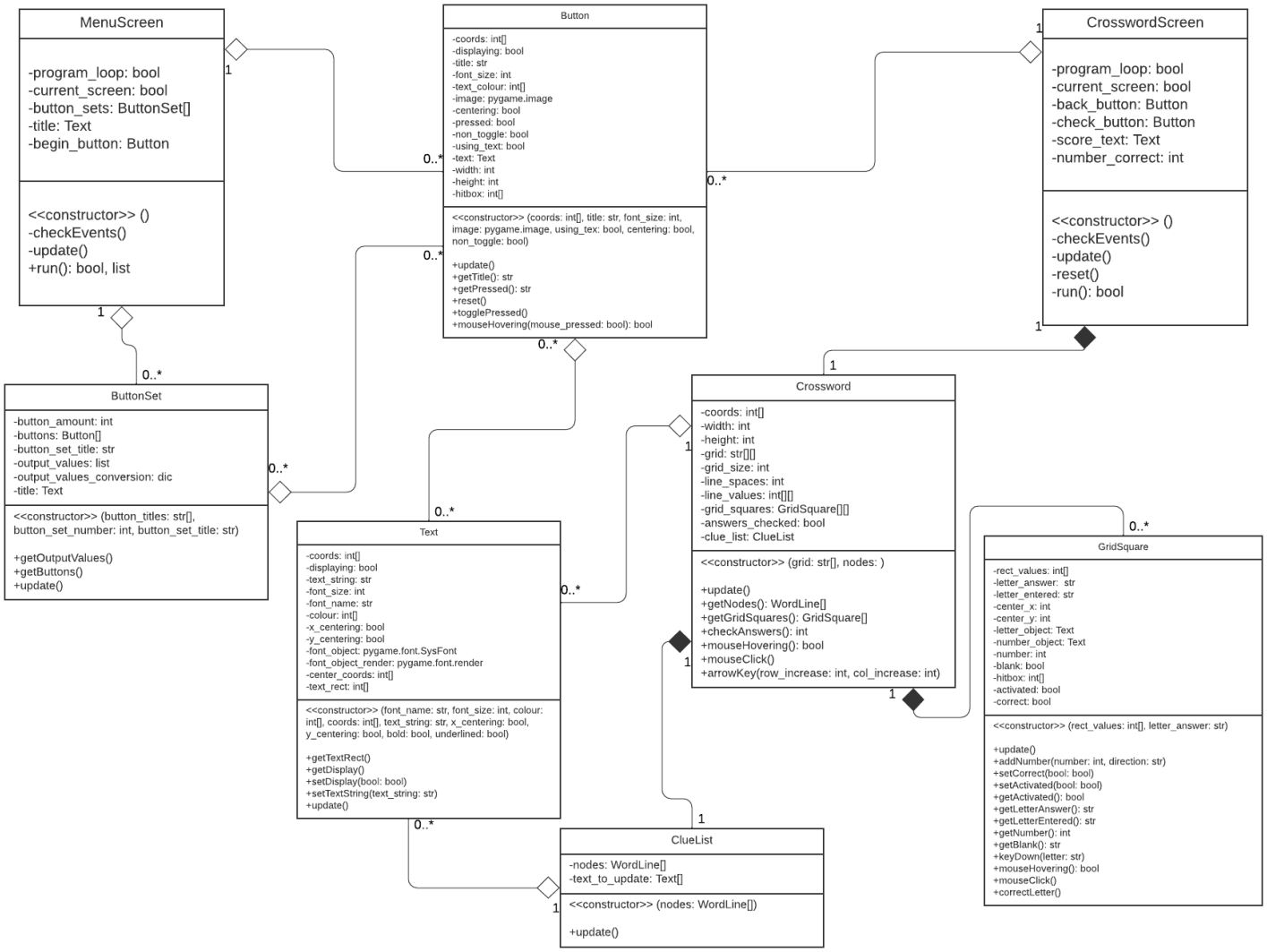


Figure 10: A UML class definition diagram for the interface program.

3 Technical Solution

3.1 main.py

```
1  import pygame
2  from random import randint, choice
3  import time
4  from crossword_generator import main
5
6  #RGB values
7  WHITE = (255, 255, 255)
8  BLACK = (0, 0, 0)
9  RED = (255, 0, 0)
10 GREEN = (0, 255, 0)
11 GREY = (220,220,220)
12 BLUE = (30, 144, 255)
13
14 SCREEN_SIZE_X = 1350
15 SCREEN_SIZE_Y = 810
16 FPS = 60
17
18 pygame.init()
19
20 fps_clock = pygame.time.Clock()
21
22 screen = pygame.display.set_mode([SCREEN_SIZE_X, SCREEN_SIZE_Y])
23 pygame.display.set_caption("Crossoword Generator")
24
25 #loads the images to be used
26 button_sprite_1 = pygame.image.load("button_4.png")
27 button_sprite_2 = pygame.image.load("button_8.png")
28 button_sprite_3 = pygame.image.load("button_9.png")
29
30
31 #A class which handles the list of clues displayed to the right of the crossword.
32
33 class ClueList():
34
35   def __init__(self, nodes):
36
37     self.__nodes = nodes
38
39     #Text() objects for the "across" and "down" headings of the clue list.
40     self.__across_title = Text("chalkboard", 35, BLACK, [SCREEN_SIZE_X/1.68,
           SCREEN_SIZE_Y/5.5], "Across", False, False, True, True)
41     self.__down_title = Text("chalkboard", 35, BLACK, [SCREEN_SIZE_X/1.26,
           SCREEN_SIZE_Y/5.5], "Down", False, False, True, True)
42
43     #A list of Text() objects with which a for loop goes through and updates each
           one.
44     self.__text_to_update = [self.__across_title, self.__down_title]
```

```
45
46      #These line space variables are used for the for loop and
47      #ensure that each word in the clue list has an appropiate vertical space
            between them.
48      line_space_across = self.__across_title.getTextRect()[3]
49      line_space_down = self.__down_title.getTextRect()[3]
50
51      for node in self.__nodes:
52
53        #A string in the format of "number. word" e.g. "1. pomme".
54        text_string = f"{node.grid_square_number}. {node.getClueWord()}"
55
56        #If the word is an adjective then it needs to have its gender shown in
            brackets.
57        if node.getGender() != "":
58          text_string += f" ({node.getGender()})"
59
60        if node.getDirection() == "across":
61          text = Text("chalkboard", 20, BLACK, [self.__across_title.getCoords()[0],
            self.__across_title.getCoords()[1]+line_space_across], text_string, False,
            False)
62          line_space_across += text.getTextRect()[3]
63
64        else:
65          text = Text("chalkboard", 20, BLACK, [self.__down_title.getCoords()[0],
            self.__down_title.getCoords()[1]+line_space_down], text_string, False, False
            )
66          line_space_down += text.getTextRect()[3]
67
68        self.__text_to_update.append(text)
69
70    def update(self):
71
72      for text in self.__text_to_update:
73        text.update()
74
75
76  #A class which creates objects for each square in the Crossword().
77
78  class GridSquare():
79
80    def __init__(self, rect_values, letter_answer):
81
82      #A list in the format (x_pos, y_pos, width, height) which contains the
            neccessary details to draw a rectangle.
83      self.__rect_values = rect_values
84
85      #Stores the letter which this grid square should contain.
86      self.__letter_answer = letter_answer
87
88      #Stores the letter which the user has entered.
```

```
89      self.__letter_entered = ""
90
91      #The little word-line number which may be displayed in the top left-hand corner
            if this is the first square of a word.
92      self.__number = 0
93
94      #A letter_answer of "X" indicates that this is a blank square which should be
            filled with black.
95      if self.__letter_answer == "X":
96        self.__blank = True
97      else:
98        self.center_x = (self.__rect_values[0] + self.__rect_values[2] / 2)
99        self.center_y = (self.__rect_values[1] + self.__rect_values[3] / 2)
100       self.__letter_object = ""
101       self.__number_object = ""
102       self.__blank = False
103
104     #A list in the format [(start_x, end_x), (start_y, end_y)] which shows the
            range of pixels which this rectangle governs.
105     self.__hitbox = [(self.__rect_values[0], self.__rect_values[0] + self.
            __rect_values[2]), (self.__rect_values[1], self.__rect_values[1] + self.
            __rect_values[3])]
106
107     #A boolean to show whether the user has clicked on this square to type into it.
108     self.__activated = False
109
110     #If the letter in this square is the correct letter.
111     self.__correct = True
112
113   def getActivated(self):
114     return self.__activated
115
116   def getLetterAnswer(self):
117     return self.__letter_answer
118
119   def getLetterEntered(self):
120     return self.__letter_entered
121
122   def getNumber(self):
123     return self.__number
124
125   def getBlank(self):
126     return self.__blank
127
128   def setCorrect(self, bool):
129     self.__correct = bool
130
131   def setActivated(self, bool):
132     self.__activated = bool
133
134   def addNumber(self, number):
```

```
135
136       self.__number = number
137       self.__number_object = Text("chalkboard", 15, BLUE, [self.__rect_values[0]+4,
              self.__rect_values[1]+3], str(self.__number), False, False)
138
139     def keyDown(self, letter):
140
141       if self.__activated:
142         self.__letter_entered = letter
143         self.__letter_object = Text("chalkboard", 25, BLACK, [self.center_x, self.
            center_y], self.__letter_entered)
144
145     def mouseClick(self):
146
147       self.__activated = self.mouseHovering()
148
149
150     #A method which returns true if the user's mouse lies within this square.
151     #A boolean mousePressed is passed through to indicate that the user has also
            clicked their mouse.
152     def mouseHovering(self):
153
154       mouse_x = pygame.mouse.get_pos()[0]
155       mouse_y = pygame.mouse.get_pos()[1]
156
157       if hitBoxesTouching(self.__hitbox, [(mouse_x, mouse_x), (mouse_y, mouse_y)])
              and not self.__blank:
158         return True
159       else:
160         return False
161
162     def correctLetter(self):
163
164       if self.__correct:
165         colour = GREEN
166       else:
167         colour = RED
168
169       self.__letter_object = Text("chalkboard", 25, colour, [self.center_x, self.
              center_y], self.__letter_answer)
170
171     def update(self):
172
173       if not self.__blank:
174
175         #If activated then draw a blue rectangle to indicate to the user that this
            square is active.
176         if self.__activated:
177           pygame.draw.rect(screen, BLUE, self.__rect_values, width=3)
178
179         if self.__letter_object != "":
```

```
180           self.__letter_object.update()
181         if self.__number_object != "":
182           self.__number_object.update()
183 
184       else:
185         #A square filled with black.
186         pygame.draw.rect(screen, BLACK, self.__rect_values)
187 
188 
189 #A class which represents the crossword.
190 
191 class Crossword():
192 
193   def __init__(self, grid, nodes):
194 
195     #The 2D array generated by crossword_generator.py and passed through in the
          __init__ method.
196     self.__grid = grid
197 
198     #The nodes passed through from crossword_generator.py.
199     self.__nodes = nodes
200 
201     self.__grid_size = len(self.__grid)
202     self.__height = SCREEN_SIZE_Y / 1.5
203     self.__width = self.__height
204     self.__coords = [SCREEN_SIZE_X/10, SCREEN_SIZE_Y/5.5]
205 
206     #The number of pixels between each GridSquare().
207     self.__line_spaces = self.__width / self.__grid_size
208 
209     #A list of tuples in the format [((start_x, start_y), (end_x, end_y)), ...]
210     #which contain the neccessary details to draw all of the lines.
211     self.__line_values = []
212 
213     #A 2D array which will contain the GridSquare() objects.
214     self.__grid_squares = [[] for _ in range(self.__grid_size)]
215 
216     #A boolean which indicates whether the check-answers button has been pressed.
217     self.__answers_checked = False
218 
219     #This chunk of code fills the line_values list with all of the neccessary
          values.
220     vertical_line_space = 0
221     horizontal_line_space = 0
222     for i in range(self.__grid_size + 1):
223 
224       self.__line_values.append(((self.__coords[0] + vertical_line_space, self.
          __coords[1]),
225                                  (self.__coords[0] + vertical_line_space, self.
          __coords[1] + self.__height)))
226       vertical_line_space += self.__line_spaces
```

```
227
228         self.__line_values.append(((self.__coords[0], self.__coords[1] +
              horizontal_line_space),
229                                    (self.__coords[0] + self.__width, self.__coords[1] +
              horizontal_line_space)))
230         horizontal_line_space += self.__line_spaces
231
232       #Fills the grid_squares 2D array.
233       for row in range(len(self.__grid)):
234         for col in range(len(self.__grid[row])):
235
236           #Uses self.line_spaces to indicate how wide and high the rectangle should
              be.
237           #2 is added to account for the line width.
238           rect_values = [self.__coords[0] + (self.__line_spaces * col), self.__coords
              [1] + (self.__line_spaces * row), self.__line_spaces + 2, self.__line_spaces
               + 2]
239
240           #Puts the GridSquare() object into the correct row of the 2D array.
241           self.__grid_squares[row].append(GridSquare(rect_values, grid[row][col]))
242
243       #This for loop adds numbers to the GridSquare() objects where neccessary.
244       for node in self.__nodes:
245
246         if self.__grid_squares[node.getStartRow()][node.getStartCol()].getNumber() !=
              0:
247           self.__grid_squares[node.getStartRow()][node.getStartCol()].addNumber(self.
            __grid_squares[node.getStartRow()][node.getStartCol()].getNumber())
248         else:
249           self.__grid_squares[node.getStartRow()][node.getStartCol()].addNumber(node.
            getNumber())
250
251         node.grid_square_number = self.__grid_squares[node.getStartRow()][node.
            getStartCol()].getNumber()
252
253       self.__clue_list = ClueList(self.__nodes)
254
255     def getNodes(self):
256       return self.__nodes
257
258     def getGridSquares(self):
259       return self.__grid_squares
260
261     #A method which is ran if one of the arrow keys is pressed and shifts which
          GridSquare() is activated.
262
263     def arrowKey(self, row_increase, col_increase):
264
265       #This chunk finds which GridSquare is currently activated.
266       a_grid_activated = False
267       for row in range(len(self.__grid_squares)):
```

```
268         for col, grid_square in enumerate(self.__grid_squares[row]):
269           if grid_square.getActivated():
270             a_grid_activated = True
271             grid_square_row = row
272             grid_square_col = col
273             grid_square_activated = grid_square
274 
275       #This chunk changes the activated GridSquare() to one of the adjacent ones
              depending on which arrow key is pressed.
276       if a_grid_activated:
277         new_row = grid_square_row + row_increase
278         new_col = grid_square_col + col_increase
279         if (new_row < len(self.__grid_squares)) and (new_col < len(self.
            __grid_squares[grid_square_col])):
280           if not self.__grid_squares[new_row][new_col].getBlank():
281             grid_square_activated.setActivated(False)
282             self.__grid_squares[new_row][new_col].setActivated(True)
283 
284 
285     #This method checks which word-lines and letters are correct and returns the
          number of correct answers.
286 
287     def checkAnswers(self):
288 
289       self.__answers_checked = True
290 
291       for node in self.__nodes:
292 
293         #Assumes the node is correct.
294         node.setCorrect(True)
295 
296         if node.getDirection() == "down":
297 
298           for row in range(node.getStartRow(), node.getEndRow()+1):
299             #Compares the letter in the grid square to the corresponing letter in the
               expected answer provided by the node.
300             if self.__grid_squares[row][node.getStartCol()].getLetterEntered() !=
              node.getFillingInWord()[row - node.getStartRow()]:
301               node.setCorrect(False)
302               self.__grid_squares[row][node.getStartCol()].setCorrect(False)
303               self.__grid_squares[row][node.getStartCol()].correctLetter()
304         else:
305 
306           for col in range(node.getStartCol(), node.getEndCol()+1):
307             if self.__grid_squares[node.getStartRow()][col].getLetterEntered() !=
              node.getFillingInWord()[col - node.getStartCol()]:
308               node.setCorrect(False)
309               self.__grid_squares[node.getStartRow()][col].setCorrect(False)
310               self.__grid_squares[node.getStartRow()][col].correctLetter()
311 
312       num_correct = 0
```

```
313       for node in self.__nodes:
314         if node.getCorrect():
315           num_correct += 1
316 
317       return num_correct
318 
319     def mouseHovering(self):
320 
321       hovering = False
322 
323       for row in range(len(self.__grid_squares)):
324         for square in self.__grid_squares[row]:
325 
326           if square.mouseHovering():
327             hovering = True
328 
329       return hovering
330 
331     def mouseClick(self):
332 
333       for row in range(len(self.__grid_squares)):
334         for square in self.__grid_squares[row]:
335 
336           square.mouseClick()
337 
338     def update(self):
339 
340       #Draws all of the lines.
341       for coord_set in self.__line_values:
342         pygame.draw.line(screen, BLACK, coord_set[0], coord_set[1], width = 2)
343 
344       #updates_last is used because the blue activated rectangle needs to be
             displayed above the crossword
345       update_last = ""
346 
347       for row in range(len(self.__grid_squares)):
348         for grid_square in self.__grid_squares[row]:
349 
350           if grid_square.getActivated():
351             update_last = grid_square
352           else:
353             grid_square.update()
354 
355       if update_last != "":
356         update_last.update()
357 
358       self.__clue_list.update()
359 
360 
361 #A class which handles every bit of text used in the interface.
362 
```

```
363 class Text():
364
365   def __init__(self, font_name, font_size, colour, coords, text_string, x_centering
          =True, y_centering=True, bold=False, underlined=False):
366
367     self.__text_string = text_string
368     self.__colour = colour
369     self.__coords = coords
370     self.__font_name = font_name
371     self.__font_size = font_size
372
373     self.__displaying = True
374     self.__x_centering = x_centering
375     self.__y_centering = y_centering
376     self.__bold = bold
377     self.__underlined = underlined
378
379     self.update()
380
381   def setColour(self, colour):
382     self.__colour = colour
383
384   def setDisplay(self, bool):
385     self.__displaying = bool
386
387   def setTextString(self, text_string):
388     self.__text_string = text_string
389
390   def getTextRect(self):
391     return self.__text_rect
392
393   def getDisplaying(self):
394     return self.__displaying
395
396   def getCoords(self):
397     return self.__coords
398
399   def update(self):
400
401     #Intialise the font_object.
402     self.font_object = pygame.font.SysFont(self.__font_name, self.__font_size, bold
          =self.__bold)
403     if self.__underlined:
404       self.font_object.set_underline(True)
405
406     #Render the font_object.
407     self.font_object_render = self.font_object.render(self.__text_string, True,
          self.__colour)
408
409     #Get the coords of the center of the text.
410     self.center_coords = self.font_object_render.get_rect(center=self.__coords)
```

```
411
412     #This chunk creates a list called text_rect to represent a box around the text
413     #in the format (x_pos, y_pos, width, height). The box is 15 pixels larger than
            the text.
414     text_rect_font_object = pygame.font.SysFont(self.__font_name, self.__font_size
            + 15, bold=self.__bold)
415     text_rect_width, text_rect_height = text_rect_font_object.size(self.
            __text_string)
416     text_rect_font_object = text_rect_font_object.render(self.__text_string, True,
            self.__colour)
417     self.__text_rect = [text_rect_font_object.get_rect(center=self.__coords)[0],
            text_rect_font_object.get_rect(center=self.__coords)[1], text_rect_width,
            text_rect_height]
418
419     #This chunk deals with centering.
420     if not self.__x_centering:
421       self.center_coords[0] = self.__coords[0]
422       self.__text_rect[0] = self.__coords[0]
423     if not self.__y_centering:
424       self.center_coords[1] = self.__coords[1]
425       self.__text_rect[1] = self.__coords[1]
426
427     #Finally, display the text on the screen.
428     if self.__displaying:
429       screen.blit(self.font_object_render, self.center_coords)
430
431
432 #A class which represents a collection of Button() objects.
433
434 class ButtonSet():
435
436   def __init__(self, button_titles, button_set_number, button_set_title):
437
438     self.__button_amount = len(button_titles)
439     self.__buttons = []
440     self.__button_set_title = button_set_title
441
442     #These are the values outputed once the options have been selected by the user.
443     self.__output_values = []
444
445     #This is a dictionary which converts the button titles to appropiate output
            values for the crossword_generator.py.
446     self.__output_values_conversion = {"Unit 1":"diversite", "Unit 2":"marginalise"
            , "Unit 3":"criminels", "Unit 4":"",
447                   "8x8":8, "10x10":10, "12x12":12, "14x14":14, "French":"French"
            , "English":"English"}
448
449     x_val = SCREEN_SIZE_X / 3
450     y_val = (SCREEN_SIZE_Y / 5) * button_set_number
451
```

```
452       self.__title = Text("chalkboard", 27, BLUE, (30, y_val), self.
              __button_set_title, False, True)
453
454       #Creates the Button() objects for this ButtonSet().
455       for i in range(self.__button_amount):
456         self.__buttons.append(Button((x_val, y_val), button_titles[i]))
457         x_val += SCREEN_SIZE_X/5
458
459     def getOutputValues(self):
460       return self.__output_values
461
462     def getButtons(self):
463       return self.__buttons
464
465     def updateObjects(self):
466
467       for i, button in enumerate(self.__buttons):
468
469         button.update()
470
471         #This chunk decides what the output_values are.
472         if button.getPressed() and self.__output_values_conversion[button.getTitle()]
                not in self.__output_values:
473           self.__output_values.append(self.__output_values_conversion[button.getTitle
              ()])
474         elif not button.getPressed() and self.__output_values_conversion[button.
              getTitle()] in self.__output_values:
475           self.__output_values.remove(self.__output_values_conversion[button.getTitle
              ()])
476
477       self.__title.update()
478
479
480   #A class for all the buttons.
481
482   class Button():
483
484     def __init__(self, coords, title, font_size = 18, image=button_sprite_1,
              using_text=True, centering=True, non_toggle=False):
485
486       self.__coords = coords
487       self.__displaying = True
488       self.__title = title
489       self.__font_size = font_size
490       self.__text_colour = BLACK
491       self.__image = image
492       self.__centering = centering
493       self.__pressed = False
494
495       #This boolean indicates that the button will disappear once it is pressed.
496       self.__non_toggle = non_toggle
```

```
497
498       #This boolean indicates whether the button has the need for a Text() object for
              its title.
499       self.__using_text = using_text
500
501       self.__text = Text("chalkboard", self.__font_size, self.__text_colour, self.
              __coords, self.__title, self.__centering, self.__centering)
502
503       self.__width, self.__height = self.__text.getTextRect()[2], self.__text.
              getTextRect()[3]
504
505       self.__hitbox = [(self.__text.getTextRect()[0], self.__text.getTextRect()[0] +
              self.__width), (self.__text.getTextRect()[1], self.__text.getTextRect()[1] +
               self.__height)]
506
507     def mouseHovering(self, mouse_pressed):
508
509       if self.__displaying:
510
511         mouse_x = pygame.mouse.get_pos()[0]
512         mouse_y = pygame.mouse.get_pos()[1]
513
514         if hitBoxesTouching(self.__hitbox, [(mouse_x, mouse_x), (mouse_y, mouse_y)]):
515
516           if mouse_pressed:
517             self.togglePressed()
518           return True
519
520         else:
521           return False
522
523     def reset(self):
524
525       self.__displaying = True
526       self.__pressed = False
527
528     def getTitle(self):
529       return self.__title
530
531     def getPressed(self):
532       return self.__pressed
533
534     def togglePressed(self):
535
536       if not self.__non_toggle:
537
538         if self.__pressed:
539           self.__pressed = False
540           self.__text.setColour(BLACK)
541         else:
542           self.__pressed = True
```

```
543           self.__text.setColour(GREEN)
544
545       else:
546
547         self.__pressed = True
548         self.__displaying = False
549
550     def update(self):
551
552       if self.__displaying:
553
554         screen.blit(pygame.transform.smoothscale(self.__image, (self.__width, self.
            __height)), (self.__text.getTextRect()[0], self.__text.getTextRect()[1]))
555
556         if self.__using_text:
557           self.__text.update()
558
559
560 #A class for the menu screen.
561
562 class MenuScreen():
563
564     def __init__(self):
565
566       self.__program_loop = True
567
568       #A boolean which indicates whether to display this screen or not.
569       self.__current_screen = False
570
571       self.__button_sets = [ButtonSet(["8x8","10x10","12x12","14x14"], 1, "Select
              Grid Dimension(s):"),
572                             ButtonSet(["Unit 1","Unit 2","Unit 3"], 2, "Select Topic(s):"),
573                             ButtonSet(["French","English"], 3, "Select a Language(s):")]
574
575       #A Text() object for the title of the MenuScreen().
576       self.__title = Text("chalkboard", 52, BLUE, (SCREEN_SIZE_X/2, 30), "French
              Crossword Game", bold=True, underlined=True)
577
578       #A Button() object for the button you use to submit your options.
579       self.__begin_button = Button([SCREEN_SIZE_X/2, SCREEN_SIZE_Y/1.1], "Create
              Crossword", 20, button_sprite_2, False)
580
581
582     # A method which checks and handles all of the possible events and user inputs.
583
584     def __checkEvents(self):
585
586       #This chunk checks if the mouse is hovering over something clickable.
587       hovering = False
588       for button_set in self.__button_sets:
589         for button in button_set.getButtons():
```

```
590         if button.mouseHovering(False):
591           hovering = True
592     if self.__begin_button.mouseHovering(False):
593       hovering = True
594 
595     #Change the mouse to its "hand" form for if hovering is True.
596     if hovering:
597       pygame.mouse.set_cursor(pygame.SYSTEM_CURSOR_HAND)
598     else:
599       pygame.mouse.set_cursor(pygame.SYSTEM_CURSOR_ARROW)
600 
601     #Goes through all of the current input events from the user.
602     for event in pygame.event.get():
603 
604       #If the user presses the red exit button of the window then quit the program.
605       if event.type == pygame.QUIT:
606         self.__current_screen = False
607         self.__program_loop = False
608 
609       #If the mouse is clicked then check if it impacts a button.
610       elif event.type == pygame.MOUSEBUTTONDOWN:
611         for button_set in self.__button_sets:
612           for button in button_set.getButtons():
613             button.mouseHovering(True)
614         self.__begin_button.mouseHovering(True)
615 
616     #If the begin_button has been pressed then change to the CrosswordScreen().
617     if self.__begin_button.getPressed():
618       self.__begin_button.togglePressed()
619       self.__current_screen = False
620 
621   def __updateObjects(self):
622 
623     screen.fill(GREY)
624 
625     self.__title.update()
626 
627     for button_set in self.__button_sets:
628       button_set.updateObjects()
629 
630     self.__begin_button.update()
631 
632     pygame.display.update()
633     fps_clock.tick(FPS)
634 
635 
636   #A method which loops while this object is the screen.
637 
638   def run(self):
639 
640     self.__current_screen = True
```

```
641
642     while self.__current_screen:
643
644       self.__checkEvents()
645
646       self.__updateObjects()
647
648     output_values = []
649     for button_set in self.__button_sets:
650       output_values.append(button_set.getOutputValues())
651
652     return self.__program_loop, output_values
653
654
655 #A class for the crossword screen.
656
657 class CrosswordScreen():
658
659   def __init__(self):
660
661     self.__current_screen = False
662     self.__program_loop = True
663
664     #A Button() object for the button which returns you to the MenuScreen().
665     self.__back_button = Button([0, 0], "Back Button", 20, button_sprite_3, False,
          False)
666
667     #A Button() object for the non-toggle button which you press to check your
          answer.
668     self.__check_button = Button([SCREEN_SIZE_X - 120, 20], "Check Crossword",
          non_toggle=True)
669
670     #The text which displays the user's score once they have checked their answers.
671     self.__score_text = Text("chalkboard", 52, BLUE, (SCREEN_SIZE_X - 100, 30), "")
672     self.__score_text.displaying = False
673
674
675   #A method which resets some of the CrossWordScreen() attributes once you have
          finished with the current crossword.
676
677   def __reset(self):
678
679     self.__number_correct = -1
680     self.__check_button.reset()
681     self.__score_text.setDisplay(False)
682
683   def __checkEvents(self):
684
685     hovering = self.crossword.mouseHovering()
686
```

```
687     if self.__back_button.mouseHovering(False) or self.__check_button.mouseHovering
            (False):
688       hovering = True
689 
690     if hovering:
691       pygame.mouse.set_cursor(pygame.SYSTEM_CURSOR_HAND)
692     else:
693       pygame.mouse.set_cursor(pygame.SYSTEM_CURSOR_ARROW)
694 
695     #If the check_button is pressed then check the answers and display the
            score_text.
696     if self.__check_button.getPressed() and not self.__score_text.getDisplaying():
697       self.__number_correct = self.crossword.checkAnswers()
698 
699       self.__score_text.setTextString(f"{self.__number_correct}/{len(self.crossword
            .getNodes())}")
700       self.__score_text.setDisplay(True)
701 
702     for event in pygame.event.get():
703 
704       if event.type == pygame.QUIT:
705         self.__current_screen = False
706         self.__program_loop = False
707 
708       elif event.type == pygame.MOUSEBUTTONDOWN:
709 
710         self.__back_button.mouseHovering(True)
711         self.__check_button.mouseHovering(True)
712         self.crossword.mouseClick()
713 
714       elif event.type == pygame.KEYDOWN:
715 
716         row_increase = 0
717         col_increase = 0
718         if event.key == pygame.K_UP:
719           row_increase = -1
720           self.crossword.arrowKey(row_increase, col_increase)
721         elif event.key == pygame.K_DOWN:
722           row_increase = 1
723           self.crossword.arrowKey(row_increase, col_increase)
724         elif event.key == pygame.K_LEFT:
725           col_increase = -1
726           self.crossword.arrowKey(row_increase, col_increase)
727         elif event.key == pygame.K_RIGHT:
728           col_increase = 1
729           self.crossword.arrowKey(row_increase, col_increase)
730 
731         #This handles if a letter is typed into a GridSquare().
732         elif event.unicode != "":
733 
734           #Checks if it is a lower-case letter of the alphabet.
```

```
735             if ord(event.unicode) >= 97 and ord(event.unicode) <= 122:
736 
737               for row in range(len(self.crossword.getGridSquares())):
738                 for square in self.crossword.getGridSquares()[row]:
739 
740                   square.keyDown(event.unicode)
741 
742       #If the back_button is pressed then return to the MenuScreen().
743       if self.__back_button.getPressed():
744         self.__back_button.togglePressed()
745         self.__current_screen = False
746 
747     def __updateObjects(self):
748 
749       screen.fill(WHITE)
750 
751       self.__back_button.update()
752       self.__check_button.update()
753       self.crossword.update()
754 
755       self.__score_text.update()
756 
757       pygame.display.update()
758       fps_clock.tick(FPS)
759 
760     def run(self, input_values):
761 
762       self.__reset()
763 
764       self.input_values = input_values
765 
766       if input_values[0] == []:
767         self.input_values[0] = [8, 10, 12, 14]
768       crossword_size = choice(self.input_values[0])
769 
770       if input_values[1] == []:
771         self.input_values[1] = ["diversite", "marginalise", "criminels"]
772       topics = self.input_values[1]
773 
774       if input_values[2] == []:
775         self.input_values[2] = ["French", "English"]
776       language = choice(self.input_values[2])
777 
778       self.__current_screen = True
779       grid, nodes = main(crossword_size, topics, language)
780       self.crossword = Crossword(grid, nodes)
781 
782       while self.__current_screen:
783 
784         self.__checkEvents()
785 
```

```
786         self.__updateObjects()
787 
788     return self.__program_loop
789 
790 
791 #This function compares two hitbox lists to see if they intersect with each other
          and return True if they do.
792 
793 def hitBoxesTouching(hitbox_1, hitbox_2):
794 
795     x_touching = True
796     y_touching = True
797 
798     if hitbox_1[0][0] > hitbox_2[0][1] or hitbox_1[0][1] < hitbox_2[0][0]:
799         x_touching = False
800 
801     if hitbox_1[1][1] < hitbox_2[1][0] or hitbox_1[1][0] > hitbox_2[1][1]:
802         y_touching = False
803 
804     if y_touching and x_touching:
805         return True
806     else:
807       return False
808 
809 
810  #This procedure keeps looping and switching between the menu and crossword screens
          untill the program is quitted.
811 
812 def programLoop(program_loop):
813 
814   menu_screen = MenuScreen()
815   crossword_screen = CrosswordScreen()
816 
817   while program_loop:
818 
819     program_loop, output_values = menu_screen.run()
820 
821     if program_loop:
822       program_loop = crossword_screen.run(output_values)
823 
824 if __name__ == "__main__":
825 
826   programLoop(True)
827 
828 pygame.quit()
```

3.2 crossword-generator.py

```
import sqlite3
from random import randint, shuffle
from copy import deepcopy
from crossword_grids import crosswords

#A class to represent the nodes within the Graph() class.
#The nodes can be thought of as each being a word on the crossword grid.

class WordLine:

  def __init__(self, word_line_range, n):

    #These represent the coordinates of where the word starts and ends.
    self.__start_row, self.__end_row, self.__start_col, self.__end_col =
        word_line_range

    #An identifier for each node
    self.__number = n

    #Looks at the coordinates to decide whether the word runs accross or down the
        crossword grid.
    if self.__start_row != self.__end_row:
      self.__direction = "down"
      self.__word_length = (self.__end_row - self.__start_row) + 1
    else:
      self.__direction = "across"
      self.__word_length = (self.__end_col - self.__start_col) + 1

    self.__intersections = []

    #These attributes store the strings of the words and clues (one will be French
        and one will be English).
    self.__filling_in_word, self.__clue_word = "", ""

    #Stores the result of the last query made for this node during the traversal.
    self.__query_result = []

    #Stores the index for the query_result to select the current word it is using
        during the traversal.
    self.__query_result_index = 0

    #Indicates whether it has has a query made for it yet
    self.__had_query = False

    #Indicates the gender of the word if it is an adjective
    self.__gender = ""

    #For indicating whether this word line has been filled in correctly on the
        interactive crossword grid.
```

```
45      self.__correct = False
46
47    def getNumber(self):
48      return self.__number
49
50    def getStartCol(self):
51      return self.__start_col
52
53    def getStartRow(self):
54      return self.__start_row
55
56    def getEndCol(self):
57      return self.__end_col
58
59    def getEndRow(self):
60      return self.__end_row
61
62    def getDirection(self):
63      return self.__direction
64
65    def getWordLength(self):
66      return self.__word_length
67
68    def getIntersections(self):
69      return self.__intersections
70
71    def getFillingInWord(self):
72      return self.__filling_in_word
73
74    def getClueWord(self):
75      return self.__clue_word
76
77    def getHadQuery(self):
78      return self.__had_query
79
80    def getGender(self):
81      return self.__gender
82
83    def getCorrect(self):
84      return self.__correct
85
86    def setCorrect(self, bool):
87      self.__correct = bool
88
89    #A method for the traversal.
90
91    def checkQueryIndex(self):
92
93      #Returns True if every word in the query has been tried in the traversal.
94      if self.__query_result_index >= len(self.__query_result) - 1:
95        return False
```

```
96        else:
97          return True
98
99      def addIntersection(self, WordLineConnecting, coords):
100
101       #Appends the node which this node intersects with and the coordinates where
              they intersect.
102       self.__intersections.append((WordLineConnecting, coords))
103
104     def addWord(self):
105
106       self.__filling_in_word, self.__clue_word = self.__query_result[self.
              __query_result_index][0], self.__query_result[self.__query_result_index][1]
107
108       if len(self.__query_result[self.__query_result_index]) > 3:
109         self.__gender = self.__query_result[self.__query_result_index][3]
110       else:
111         self.__gender = ""
112
113
114     #A method for the traversal.
115
116     def addQuery(self, query_result):
117
118       self.__had_query = True
119       self.__query_result_index = 0
120       self.__query_result = query_result
121
122       #suffles so that the words used in the crossword are randomised
123       shuffle(self.__query_result)
124
125       self.addWord()
126
127
128     #A method for the traversal.
129
130     def switchQueryWord(self):
131
132       #If the query_result_index has not exeeded the length of the query_result then
              incriment it.
133       if self.checkQueryIndex():
134         self.__query_result_index += 1
135         self.addWord()
136
137
138     #A method for the traversal.
139
140     def resetNode(self):
141
142       #Resets all of the attributes so that a new query_result can be tried.
143       self.__filling_in_word, self.__clue_word, self.__gender = "", "", ""
```

```
144     self.__query_result_index = 0
145     self.__query_result = []
146     self.__had_query = False
147 
148 #A class to represent the graph based off of the crossword grid.
149 
150 class Graph:
151 
152   def __init__(self, crossword, filling_in_language, topics, empty=False, debugging
          =False):
153 
154     #The empty crossword grid.
155     self.__crossword = crossword
156 
157     #A list containing all of the WordLine() objects.
158     self.__nodes = []
159 
160     #An identifier for each node.
161     self.__n = 1
162 
163     #Indicates that the graph is empty.
164     self.__empty = empty
165 
166     #A boolean which, when True, means information is printed to help with
          debugging.
167     self.__debugging = debugging
168 
169     #Can be equal to "French" or "English". It indicates which way around the
          languages are in the crossword.
170     self.__filling_in_language = filling_in_language
171 
172     #The language topics.
173     self.__topics = topics
174 
175     if self.__filling_in_language == "French":
176       self.__clue_language = "English"
177     else:
178       self.__clue_language = "French"
179 
180     self.__constructGraph(self.__crossword)
181 
182     #self.displayIntersections()
183     #self.displayNodes()
184 
185     #The try and except is used to catch recursion errors.
186     try:
187       self.__traversal(self.__nodes[0], [])
188     except Exception as e:
189       print(e)
190 
191   #A method used by __constructGraph() to find wordlines.
```

```
192
193   def __checkForSpaces(self, row, col, width, height, direction):
194
195     checking = True
196     word_line = False
197
198     while checking:
199
200       if col != width and direction == "across":
201         col+=1
202         if self.__crossword[row][col] == " ":
203           word_line = True
204         else:
205           col-=1
206           checking = False
207
208       elif row != height and direction == "down":
209         row+=1
210         if self.__crossword[row][col] == " ":
211           word_line = True
212         else:
213           row-=1
214           checking = False
215
216       else:
217         checking = False
218
219     #row is the end row of the wordline and col is the end col.
220     #word_line is a boolean that indicates whether the word_line is valid (has a
          lenght of more than 1).
221
222     return row, col, word_line
223
224
225   #A method which converts the 2D array of the empty crossword grid into a graph.
226
227   def __constructGraph(self, crossword):
228
229     #This chunk of code checks whether for each cell in the crossword grid it is
          the first cell of a wordline.
230     for row in range(len(crossword)):
231       for col in range(len(crossword[row])):
232
233         #Checks if the cell is one of the empty spaces where the letter will be
          written.
234         if crossword[row][col] == " ":
235
236           #Checks if the cell to the left isn't also an empty space.
237           if (crossword[row][col-1] != " " and col != 0) or (col == 0):
238
239             word_line_row, word_line_col, word_line = self.__checkForSpaces(row,
```

```
            col, len(crossword[row])-1, len(crossword)-1, "across")
240
241             #Checks if __checkForSpaces() has indicated that this cell is the first
                cell of a wordline.
242             if word_line:
243
244               #Adds a node to the graph which represnets this wordline.
245               self.__nodes.append(WordLine((row, word_line_row, col, word_line_col)
                , self.__n))
246               self.__n += 1
247
248           #Checks if the cell to the above isn't also an empty space.
249           if (crossword[row-1][col] != " " and row != 0) or (row == 0):
250
251             word_line_row, word_line_col, word_line = self.__checkForSpaces(row,
                col, len(crossword[row])-1, len(crossword)-1, "down")
252
253             if word_line:
254               self.__nodes.append(WordLine((row, word_line_row, col, word_line_col)
                , self.__n))
255               self.__n += 1
256
257       #This chunk finds and logs the intersections between these nodes (the
            coordinates where the words overlap thier letters).
258       for node_1 in self.__nodes:
259
260         #Goes through each coordinate of the node_1 wordline.
261         for row in range(node_1.getStartRow(), node_1.getEndRow() + 1):
262           for col in range(node_1.getStartCol(), node_1.getEndCol() + 1):
263
264             for node_2 in self.__nodes:
265
266               #Goes through each coordinate of the node_2 wordline.
267               for row_2 in range(node_2.getStartRow(), node_2.getEndRow() + 1):
268                 for col_2 in range(node_2.getStartCol(), node_2.getEndCol() + 1):
269
270                   #Checks if there is an intersection between these two nodes (they
                can't be the same node).
271                   if node_1 != node_2 and row == row_2 and col == col_2 and (node_2,
                (row, col)) not in node_1.getIntersections():
272
273                     node_1.addIntersection(node_2, (row, col))
274                     node_2.addIntersection(node_1, (row, col))
275
276
277       #A method which returns a tuple of all the filling in words attributed to the
            nodes in the graph.
278
279       def __fillingInWords(self):
280
281         words = []
```

```
282       for node in self.__nodes:
283         words.append(node.getFillingInWord())
284 
285       return tuple(words)
286 
287 
288     #A method which returns a tuple of all the clue words attributed to the nodes in
          the graph.
289 
290     def __clueWords(self):
291 
292       words = []
293       for node in self.__nodes:
294         words.append(node.getClueWord())
295 
296       return tuple(words)
297 
298 
299     #Performs a query for the traversal and returns the result.
300 
301     def __query(self, substring_values, node):
302 
303       #Creates a connection to the database.
304       with sqlite3.connect('database.db') as connection:
305 
306         cur = connection.cursor()
307 
308         if len(self.__topics) == 1:
309           self.__topics.append("filler")
310 
311         #A string that represents an SQL query.
312         query_string = f"""
313         SELECT {self.__filling_in_language}Words.{self.__filling_in_language}
            WordForCrossword, {self.__clue_language}Words.{self.__clue_language}Word,
314         FrenchWords.WordClass
315         FROM {self.__filling_in_language}Words, {self.__clue_language}Words,
            Translations
316         WHERE ({self.__filling_in_language}Words.{self.__filling_in_language}WordID =
              Translations.{self.__filling_in_language}WordID)
317         AND ({self.__clue_language}Words.{self.__clue_language}WordID = Translations
            .{self.__clue_language}WordID)
318         AND ({self.__filling_in_language}Words.WordLength = {node.getWordLength()})
319         AND ({self.__filling_in_language}Words.{self.__filling_in_language}
            WordForCrossword NOT IN {self.__fillingInWords()})
320         AND (FrenchWords.Topic IN {tuple(self.__topics)})
321         """
322 
323         #Adds all of the substring queries to the query_string.
324         for substring_value in substring_values:
325           query_string += f"AND (SUBSTR({self.__filling_in_language}WordForCrossword,
            {substring_value[0]},1) = '{substring_value[1]}')"
```

```
326
327         #Execute the query string.
328         results = cur.execute(query_string).fetchall()
329
330         #If the clues are in english, the english adjectives need the extra detail of
              what gender to translate them into.
331         if self.__filling_in_language == "French":
332
333           results = [list(x) for x in results]
334
335           for result in results:
336
337             #If the word is an adjective then perfrom a query to fetch its gender.
338             if result[2] == "adjective":
339               query_string_2 = f"""
340               SELECT FrenchGender.Gender
341               FROM FrenchGender, FrenchWords, EnglishWords, Translations
342               WHERE (FrenchWords.FrenchWordID = FrenchGender.FrenchWordID)
343               AND (FrenchWords.FrenchWordID = Translations.FrenchWordID)
344               AND (EnglishWords.EnglishWordID = Translations.EnglishWordID)
345               AND (FrenchWords.FrenchWordForCrossword = '{result[0]}')
346               """
347               result.append(cur.execute(query_string_2).fetchone()[0])
348
349     #(Debugging).
350     if self.__debugging: print(f"query result: {results}")
351
352     return results
353
354
355   #A recursive method which trys to fill in the empty grid with words through depth
          first traversal of the graph.
356
357   def __traversal(self, current_node, visited):
358
359     #A variable for storing the letters that the word must contain for the SUBSTR
          part of the query.
360     substring_values = self.__substringValues(current_node)
361
362     #A boolean which indicates whether the method will continue traversing in the
          current iteration.
363     traverse = False
364
365     #(Debgugging).
366     if self.__debugging: print(f"\ncurrent_node: {current_node.getNumber()},
          substring_values: {substring_values}")
367
368     #Runs if the current_node does not have a query result attributed to it.
369     if not current_node.checkQueryIndex() and not current_node.getHadQuery():
370
371       #(Debgugging).
```

```
372         if self.__debugging: print(f"performing query (node {current_node.getNumber()
              } does not have a query yet)")
373 
374         #Perform a query and store the results.
375         #result is a 2D array in the format ((1st word result, traslation), (2nd word
              result, traslation)).
376         result = self.__query(substring_values, current_node)
377 
378         #Checks if no results have been found for the query
379         if result == []:
380 
381           #If it is the first node being traversed and no query results are found
              then the traversal must end.
382           if visited==[]:
383 
384             #(Debugging).
385             if self.__debugging: print("End of traversal")
386 
387           #No results have been found so it returns to the previous node visited.
388           else:
389 
390             #(Debugging).
391             if self.__debugging: print(f"going back from node {current_node.getNumber
              ()} to {visited[-1].getNumber()}")
392 
393             visited_store = visited[-1]
394             visited.pop(-1)
395             self.__traversal(visited_store, visited)
396 
397         #At least one query result has been found.
398         else:
399 
400           #(Debugging).
401           if self.__debugging: print("Added query result")
402 
403           #Store the query result in the current_node object.
404           current_node.addQuery(result)
405 
406           #Continuing the depth-first traversal.
407           traverse = True
408 
409       #Runs if the current_node has a query_result but every word from that query
            result has been tried.
410       elif not current_node.checkQueryIndex() and current_node.getHadQuery():
411 
412         if visited==[]:
413 
414           if self.__debugging: print("End of traversal")
415 
416         #The current_node is reset and the program returns to the previous node
              visited.
```

```
417         else:
418 
419           current_node.resetNode()
420 
421           #(Debugging).
422           if self.__debugging:
423             print(f"reset node {current_node.getNumber()}")
424             print(f"going back from node {current_node.getNumber()} to {visited[-1].
          getNumber()}")
425 
426           visited_store = visited[-1]
427           visited.pop(-1)
428           self.__traversal(visited_store, visited)
429 
430       #Else is ran when there is a query result stored in the current_node but there
          are still some words left to try.
431       else:
432 
433         #Try the next word.
434         current_node.switchQueryWord()
435 
436         #Continuing the depth-first traversal.
437         traverse = True
438 
439       #Continue the depth-first traversal if traverse is True.
440       if traverse:
441 
442         #(Debugging).
443         if self.__debugging:
444           print("traversing")
445           print(f"selected: {current_node.getFillingInWord()}")
446 
447         #Add the current node to the visited list.
448         visited.append(current_node)
449 
450         for intersection in current_node.getIntersections():
451 
452           #(Debugging).
453           if self.__debugging: print(f"intersection: from {current_node.getNumber()}
          to {intersection[0].getNumber()}")
454 
455           #If the intersecting node has not been visited yet then visit it.
456           if intersection[0] not in visited:
457 
458             #(Debugging).
459             if self.__debugging:
460               print(f"traversing from node {current_node.getNumber()} to {
          intersection[0].getNumber()}")
461               visited_numbers = []
462               for node in visited:
463                 visited_numbers.append(node.getNumber())
```

```
464                 print(f"visited: {visited_numbers}")
465 
466             self.__traversal(intersection[0], visited)
467 
468       #(Debugging).
469       if self.__debugging:
470         visited_numbers = []
471         for node in visited:
472           visited_numbers.append(node.getNumber())
473 
474         return visited_numbers
475 
476 
477   #A method for the traversal which finds the letters that the word must contain
          for the SUBSTR part of the query.
478 
479   def __substringValues(self, node):
480 
481     substring_values = []
482 
483     #Goes through each intersection with the current node.
484     #intersections is a list of lists with the format ((intersecting node, (row of
          intersection, col of intersection)), ...)
485     for intersection in node.getIntersections():
486 
487       #Checks if the filling_in_word of the intersecting node has a word attributed
            to it yet.
488       if intersection[0].getFillingInWord() != "":
489 
490         #Finds the position of the letter (node_word_index) within the current node
            that is being intersected with.
491         if node.getDirection() == "down":
492           node_word_index = intersection[1][0] - node.getStartRow() + 1
493         else:
494           node_word_index = intersection[1][1] - node.getStartCol() + 1
495 
496         #Finds which letter is being intersected with (intersecting_letter)
497         if intersection[0].getDirection() == "down":
498           intersecting_letter_index = intersection[1][0] - intersection[0].
          getStartRow() + 1
499           intersecting_letter = intersection[0].getFillingInWord()[
          intersecting_letter_index - 1]
500         else:
501           intersecting_letter_index = intersection[1][1] - intersection[0].
          getStartCol() + 1
502           intersecting_letter = intersection[0].getFillingInWord()[
          intersecting_letter_index - 1]
503 
504         substring_values.append((node_word_index, intersecting_letter, node.
          getNumber()))
505 
```

```
506       return substring_values
507
508
509   #A method which returns a boolean for whether the graph has been completed or not
          .
510
511   def completed(self):
512
513       completed = True
514
515       for node in self.__nodes:
516         if node.getFillingInWord() == "" or node.getClueWord() == "":
517           completed = False
518
519       if self.__empty: completed = False
520
521       return completed
522
523   def displayIntersections(self):
524
525       for node in self.__nodes:
526         for vals in node.getIntersections(): print(f"node{node.getNumber()}: node{
            vals[0].getNumber()} {vals[1]}", end=", ")
527         print()
528
529   def displayNodes(self):
530
531       for node in self.__nodes:
532         print(f"Node{node.getNumber()}: (word_length: {node.getWordLength()},
            filling_in_word: {node.getFillingInWord()}, clue_word: {node.getClueWord()},
             coords ({node.getStartRow(), node.getEndRow(), node.getStartCol(), node.
            getEndCol()})")
533
534   def getNodes(self):
535       return self.__nodes
536
537
538   #A method which constructs a 2D array to represent the graph.
539
540   def getGrid(self):
541
542       self.__crossword_display = self.__crossword
543
544       for row in range(len(self.__crossword)):
545         for col in range(len(self.__crossword[row])):
546
547           for node in self.__nodes:
548
549             for row_2 in range(node.getStartRow(), node.getEndRow() + 1):
550               for col_2 in range(node.getStartCol(), node.getEndCol() + 1):
551
```

```
552                 if row == row_2 and col == col_2:
553
554                   if node.getDirection() == "down":
555                     index = row - node.getStartRow()
556                   else:
557                     index = col - node.getStartCol()
558
559                   self.__crossword_display[row][col] = node.getFillingInWord()[index]
560
561       return self.__crossword_display
562
563   def main(size, topics, language):
564
565     i = 0
566
567     shuffle(crosswords)
568
569     crossword = crosswords[i]
570
571     graph = Graph(crossword, "French", topics, empty=True)
572
573     while not graph.completed() and i < len(crosswords):
574
575       #Checks if the size of the crossword is the size you selected.
576       if len(crosswords[i]) == size:
577
578         #Construct the graph based on the crossword grid.
579         graph = Graph(deepcopy(crosswords[i]), language, topics, debugging=False)
580
581       i += 1
582
583     if graph.completed():
584
585       return graph.getGrid(), graph.getNodes()
586
587     else:
588
589       return [["fail"],["fail"]], []
590
591   if __name__ == "__main__":
592     print(main(8,["diversite","criminalite"],"French"))
```

3.3 database-compiler.py

```
import sqlite3
import unicodedata

#A function which removes unwanted articles and diacritical marks from the words.

def simplifyWord(string):

  for i in ["la ", "le ","l","l'","to "," ","un ","une "]:
    string = string.replace(i, "")

  #Gets rid of diacritical marks.
  string = ''.join(c for c in unicodedata.normalize('NFD', string) if unicodedata.
       category(c) != 'Mn')

  return string

def main():

  text_files = ["diversite", "marginalise","criminels"]

  word_class_dic = {"a":"adjective", "n":"noun", "v":"verb", "p":"phrase"}

  #connect to the database.
  with sqlite3.connect('database.db') as connection:

    #Create a cursor.
    cur = connection.cursor()

    french_primary_key = 0
    english_primary_key = 0

    #Clear any data from the last compilation of the database.
    cur.execute("DELETE FROM FrenchWords")
    cur.execute("DELETE FROM EnglishWords")
    cur.execute("DELETE FROM FrenchGender")
    cur.execute("DELETE FROM Translations")
    connection.commit()

    #Goes through each vocab file.
    for text_file in text_files:

      topic = text_file

      #Opens the file.
      with open(f"{text_file}.txt","r",encoding="utf-8") as file:

        #Goes through each line in the file.
        for line in file.readlines():

```

```
49            #Creates a list of each component.
50            line = line.replace("\n","").split(",")
51
52            #This chunk splits the line into its components.
53            french_word = line[0]
54            french_word_simplified = simplifyWord(french_word)
55            french_word_length = len(french_word_simplified)
56            if "-" not in line[1]:
57              english_words = (line[1],)
58            else:
59              english_words = line[1].split("-")
60            word_class = line[2]
61
62            #Incriments the french_primary_key so that it's unique.
63            french_primary_key += 1
64
65            cur.execute(f'''
66            INSERT INTO FrenchWords (FrenchWordID, FrenchWord, WordClass, Topic,
          WordLength, FrenchWordForCrossword)
67            VALUES ({french_primary_key}, "{french_word}", "{word_class_dic[
          word_class]}", "{topic}", {french_word_length}, "{french_word_simplified}")
          ''')
68
69            #If the word is a noun or an adjecticve then note the gender.
70            if word_class=="n" or word_class=="a":
71              gender = line[3]
72
73              cur.execute(f'''
74              INSERT INTO FrenchGender (FrenchWordID, Gender)
75              VALUES ({french_primary_key}, "{gender}")''')
76
77            #Goes through each English word (usually there's only one but sometimes
          there's two).
78            for word in english_words:
79
80              english_word_simplified = simplifyWord(word)
81              english_word_length = len(english_word_simplified)
82
83              #Checks to see if the English word already exists within the database
84              found = False
85              english_word_id = english_primary_key
86              for word_2 in cur.execute('''SELECT EnglishWord, EnglishWordID FROM
          EnglishWords''').fetchall():
87                if word_2[0] == word:
88                  found = True
89                  english_word_id = word_2[1]
90
91              #If the English word is not already in the database then add it.
92              if not found:
93
94                english_primary_key+=1
```

```
95                english_word_id = english_primary_key
96
97                cur.execute(f'''
98                INSERT INTO EnglishWords (EnglishWordID, EnglishWord, WordLength,
          EnglishWordForCrossword)
99                VALUES ({english_primary_key}, "{word}", {english_word_length}, "{
          english_word_simplified}")''')
100
101             cur.execute(f'''
102             INSERT INTO Translations (FrenchWordID, EnglishWordID)
103             VALUES ({french_primary_key}, {english_word_id})''')
104
105         #Commit the changes.
106         connection.commit()
107
108
109 def displayDatabase():
110
111   with sqlite3.connect('database.db') as connection:
112     cur = connection.cursor()
113
114     print()
115
116     for table in cur.execute("SELECT name FROM sqlite_master WHERE type='table';").
          fetchall():
117
118       print(cur.execute(f"PRAGMA table_info({table[0]});").fetchall())
119       print()
120
121     #print("-----------------------------------------------------")
122
123     for table in cur.execute("SELECT name FROM sqlite_master WHERE type='table';").
          fetchall():
124
125       values = cur.execute(f"SELECT * FROM {table[0]}").fetchall()
126       print(f"{table[0]}: {values}\n")
127
128 if __name__ == "__main__":
129   main()
130   #displayDatabase()
131
132
```

3.4 crossword-grids.py

```
crosswords = []

crosswords.append([
  [" "," "," "," "," "," "," "," "," "],
  [" ","X","X","X"," ","X","X","X","X"],
  [" ","X","X","X"," ","X","X","X","X"],
  [" ","X","X","X"," ","X","X","X","X"],
  [" "," "," "," "," "," "," "," ","X"],
  [" ","X","X","X"," ","X","X","X","X"],
  [" ","X","X","X"," ","X","X","X","X"],
  [" ","X","X","X"," ","X","X","X","X"],])

crosswords.append([
  [" "," "," "," "," "," "," "," ","X"],
  [" ","X","X","X"," ","X","X","X","X"],
  [" ","X","X","X"," ","X","X","X","X"],
  [" ","X","X","X"," ","X","X","X","X"],
  [" "," "," "," "," "," "," "," "," "],
  [" ","X","X","X"," ","X","X","X","X"],
  [" ","X","X","X"," ","X","X","X","X"],
  [" ","X","X","X"," ","X","X","X","X"],])

crosswords.append([
  ["X"," "," "," "," "," "," "," "," "],
  [" ","X","X","X","X","X"," ","X"],
  [" ","X","X","X","X","X"," ","X"],
  [" ","X","X","X","X","X"," ","X"],
  [" "," "," "," "," "," "," "," "," "],
  [" ","X","X","X","X","X"," ","X"],
  [" ","X","X","X","X","X"," ","X"],
  [" ","X","X","X","X","X"," ","X"],])

crosswords.append([
  [" "," "," "," "," "," "," "," "," "],
  ["X"," ","X","X","X","X"," ","X"],
  ["X"," ","X","X","X","X"," ","X"],
  ["X"," ","X","X","X","X"," ","X"],
  ["X"," ","X","X","X","X"," ","X"],
  ["X"," ","X","X","X","X"," ","X"],
  ["X"," ","X","X","X","X"," ","X"],
  [" "," "," "," "," "," "," "," "],])

crosswords.append([
  ["X"," ","X","X","X","X"," ","X"],
  [" "," "," "," "," "," "," "," "],
  ["X","X"," ","X","X","X"," ","X"],
  ["X","X"," ","X","X","X"," ","X"],
  ["X","X"," ","X","X","X"," ","X"],
  ["X","X"," ","X","X","X"," ","X"],
```

```
  ["X","X"," ","X","X","X"," ","X"],
  [" "," "," "," "," "," "," "," "],])

crosswords.append([
  [" "," "," "," "," "," "," "," "," "," "," "," "],
  ["X","X","X","X","X"," ","X","X"," ","X"],
  ["X","X"," ","X","X"," ","X","X"," ","X"],
  ["X","X"," ","X","X"," ","X","X"," ","X"],
  ["X","X"," ","X","X"," ","X","X"," ","X"],
  ["X","X"," ","X","X"," ","X","X"," ","X"],
  ["X"," "," "," "," "," "," "," "," ","X"],
  ["X","X"," ","X","X","X","X","X"," ","X"],
  ["X","X"," ","X","X","X","X","X"," ","X"],
  ["X","X"," ","X","X","X","X","X"," ","X"]])

crosswords.append([
  [" "," "," "," "," "," "," "," "," "," "," "," "],
  ["X","X","X","X","X"," ","X","X"," ","X"],
  ["X","X"," ","X","X"," ","X","X"," ","X"],
  ["X","X"," ","X","X"," ","X","X"," ","X"],
  ["X","X"," ","X","X"," ","X","X"," ","X"],
  ["X","X"," ","X","X"," ","X","X"," ","X"],
  ["X","X"," "," "," "," "," "," "," ","X"],
  ["X","X"," ","X","X"," ","X","X"," ","X"],
  ["X","X"," ","X","X"," ","X","X"," ","X"],
  ["X","X"," ","X","X"," ","X","X"," ","X"],])

crosswords.append([
  ["X","X","X","X","X"," ","X","X","X","X"],
  ["X","X"," ","X","X"," ","X","X"," ","X"],
  ["X","X"," ","X","X"," ","X","X"," ","X"],
  ["X","X"," ","X","X"," ","X","X"," ","X"],
  ["X","X"," ","X","X"," ","X","X"," ","X"],
  ["X","X"," ","X","X"," ","X","X"," ","X"],
  ["X","X"," "," "," "," "," "," "," ","X"],
  ["X","X"," ","X","X"," ","X","X"," ","X"],
  ["X","X"," ","X","X"," ","X","X"," ","X"],
  ["X"," "," "," "," "," "," "," "," ","X"]])

crosswords.append([
  [" "," "," "," "," "," "," "," "," "," "," ","X"],
  [" ","X","X","X","X"," ","X","X","X","X"],
  [" ","X"," ","X","X"," ","X","X","X","X"],
  [" ","X"," ","X","X"," ","X","X","X","X"],
  [" ","X"," ","X","X"," ","X","X","X","X"],
  [" ","X"," ","X","X"," ","X","X","X","X"],
  [" ","X"," "," "," "," "," "," "," ","X"],
  [" ","X"," ","X","X"," ","X","X","X","X"],
  [" ","X"," ","X","X"," ","X","X","X","X"],
  [" ","X"," ","X","X"," ","X","X","X","X"]])

```

```
crosswords.append([
  [" "," "," "," "," "," "," "," "," ","X","X"],
  ["X","X","X","X","X","X","X","X"," ","X","X"],
  ["X","X"," "," "," "," "," "," "," "," ","X"],
  ["X","X"," ","X","X","X","X"," "," ","X","X"],
  ["X","X"," ","X","X","X","X"," "," ","X","X"],
  ["X","X"," ","X","X","X","X"," "," ","X","X"],
  ["X","X"," "," "," "," "," "," "," "," ","X"],
  ["X","X"," ","X","X","X","X"," "," ","X","X"],
  ["X","X"," ","X","X","X","X"," "," ","X","X"],
  ["X","X"," ","X","X","X","X"," "," ","X","X"]])

crosswords.append([
  [" "," "," "," "," "," "," "," "," "," "," "," "," "," "],
  [" ","X","X","X"," ","X","X","X","X","X","X","X","X","X"],
  [" ","X","X","X"," ","X","X"," ","X","X"," ","X"],
  [" ","X","X","X"," ","X","X"," ","X","X"," ","X"],
  [" ","X","X","X"," ","X","X"," ","X","X"," ","X"],
  [" ","X","X","X"," ","X","X"," ","X","X"," ","X"],
  [" ","X"," "," "," "," "," "," "," "," "," "," "," ","X"],
  [" ","X","X","X","X","X","X","X"," ","X","X"," ","X"],
  [" ","X","X","X","X","X","X","X"," ","X","X"," ","X"],
  [" ","X","X","X","X","X","X","X"," ","X","X"," ","X"],
  [" ","X","X","X","X","X","X","X"," ","X","X"," ","X"]])

crosswords.append([
  [" "," "," "," "," "," "," ","X","X","X","X","X","X"],
  ["X","X","X"," ","X"," "," "," "," "," "," "," "," "],
  ["X","X","X"," ","X"," ","X","X","X","X","X","X"],
  ["X","X"," "," "," "," "," "," "," "," "," "," "," "],
  ["X","X","X"," ","X"," ","X","X"," ","X","X","X"],
  ["X","X","X"," ","X"," ","X","X"," ","X","X","X"],
  ["X","X","X"," ","X"," ","X","X"," ","X","X","X"],
  ["X","X","X"," ","X"," ","X","X"," ","X","X","X"],
  ["X","X","X"," ","X"," ","X","X"," ","X","X","X"],
  ["X","X","X"," ","X"," ","X","X"," ","X","X","X"],
  ["X","X","X"," ","X"," ","X","X"," ","X","X","X"]])

crosswords.append([
  ["X","X","X","X","X","X","X","X","X","X","X","X"],
  ["X","X"," "," "," "," "," "," "," "," "," ","X","X"],
  ["X","X"," ","X","X"," ","X","X","X"," ","X","X"],
  ["X","X"," ","X","X"," ","X"," ","X"," ","X","X"],
  ["X","X"," ","X","X"," ","X"," ","X"," ","X","X"],
  ["X","X"," ","X","X"," ","X"," ","X"," ","X","X"],
  ["X","X"," ","X","X"," ","X"," ","X"," ","X","X"],
  ["X","X"," ","X","X"," ","X"," ","X"," ","X","X"],
  ["X","X"," ","X","X"," ","X"," ","X"," ","X","X"],
  ["X","X"," ","X","X"," ","X"," ","X"," ","X","X"],
```

```
    ["X","X"," ","X","X"," ","X"," ","X"," ","X","X"],
    ["X","X"," "," "," "," "," "," "," "," ","X","X"]])

crosswords.append([
    ["X","X","X","X","X","X","X","X","X","X","X","X"],
    ["X","X","X"," "," "," "," "," "," "," ","X","X"],
    ["X","X"," ","X","X"," ","X"," ","X"," ","X","X"],
    ["X","X"," ","X","X"," ","X"," ","X"," ","X","X"],
    ["X","X"," ","X","X"," ","X"," ","X"," ","X","X"],
    ["X","X"," ","X","X"," ","X"," ","X"," ","X","X"],
    ["X","X"," ","X","X"," ","X"," ","X"," ","X","X"],
    ["X","X"," ","X","X"," ","X"," ","X"," ","X","X"],
    ["X","X"," ","X","X"," ","X"," ","X"," ","X","X"],
    ["X","X"," ","X","X"," ","X"," ","X"," ","X","X"],
    ["X","X"," "," "," "," "," "," "," "," ","X","X"]])

crosswords.append([
    [" "," "," "," "," "," "," "," "," "," "," "," "," "," "],
    ["X"," ","X","X","X","X","X","X","X","X"," ","X"],
    ["X"," ","X","X","X","X","X","X","X","X"," ","X"],
    ["X"," ","X","X","X","X","X","X","X","X"," ","X"],
    ["X"," ","X","X","X","X","X","X","X","X"," ","X"],
    ["X"," "," "," "," "," "," "," "," "," "," ","X"],
    ["X"," ","X","X","X","X","X","X","X","X"," ","X"],
    ["X"," ","X","X","X","X","X","X","X","X"," ","X"],
    ["X"," "," "," "," "," "," "," "," ","X"," ","X"],
    ["X"," ","X","X","X","X","X","X","X","X"," ","X"],
    ["X"," ","X","X","X","X","X","X","X","X"," ","X"],
    [" "," "," "," "," "," "," "," "," "," "," "," "," "," "]])

crosswords.append([
    [" "," "," "," "," "," "," "," "," "," "," "," "," "," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X"],
    [" ","X","X","X"," ","X","X"," ","X"," ","X","X"],
    [" ","X","X","X"," ","X","X"," ","X"," ","X","X"],
    [" ","X","X","X"," ","X","X"," ","X"," ","X","X"],
    [" ","X","X","X"," ","X","X"," ","X"," ","X","X"],
    [" ","X","X","X"," "," "," "," "," "," "," "," "],
    [" ","X","X","X"," ","X","X"," ","X"," ","X","X"],
    [" ","X","X","X"," ","X","X"," ","X"," ","X","X"],
    [" ","X","X","X"," ","X","X"," ","X"," ","X","X"],
    [" ","X","X","X"," ","X","X"," ","X"," ","X","X"],
    [" ","X","X","X","X","X","X","X"," ","X","X","X"]])

crosswords.append([
    ["X"," "," "," "," "," "," "," "," "," "," ","X","X"],
    ["X","X"," ","X","X","X","X","X","X"," ","X","X"],
    ["X","X"," ","X","X","X","X"," ","X"," ","X","X"],
    ["X"," "," "," "," "," "," "," "," ","X"," ","X","X"],
    ["X","X"," ","X","X","X","X"," ","X"," ","X","X"],
```

```
    ["X","X"," ","X","X","X","X"," ","X"," ","X","X"],
    ["X","X"," ","X","X","X","X"," ","X"," ","X","X"],
    ["X","X"," ","X","X","X","X"," ","X"," ","X","X"],
    ["X","X"," ","X","X","X","X"," ","X"," ","X","X"],
    ["X","X"," ","X","X","X","X"," ","X"," ","X","X"],
    ["X","X"," ","X"," "," "," "," "," "," "," "," "],
    ["X","X","X","X","X","X","X"," ","X"," ","X","X"]])

crosswords.append([
    [" "," "," "," "," "," "," "," "," "," "," "," "," "," "," "," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" "," "," "," "," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," "," "," "," "," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" "," "," "," "," "," "," "," "," "," "," "," "," "," "],
    ["X","X","X","X","X","X","X","X","X","X","X","X","X","X","X"],
    ["X","X","X","X","X","X","X","X","X","X","X","X","X","X","X"],])

crosswords.append([
    [" "," "," "," "," "," "," "," "," "," "," "," "," "," "," "," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" "," "," "," "," "," "," "," "," "," "," "," "," "," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" "," "," "," "," "," "," "," "," "," "," "," "," "," "],
    ["X","X","X","X","X","X","X","X","X","X","X","X","X","X","X"],
    ["X","X","X","X","X","X","X","X","X","X","X","X","X","X","X"],])

crosswords.append([
    [" "," "," "," "," "," "," "," "," "," "," "," "," "," "," "," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" "," "," "," "," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
    [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
```

```
  [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
  [" "," "," "," "," "," "," "," "," "," "," "," "," "," "],
  ["X","X","X","X","X","X","X","X","X","X","X","X","X","X"],
  ["X","X","X","X","X","X","X","X","X","X","X","X","X","X"],])

crosswords.append([
  [" "," "," "," "," "," "," "," "," "," "," "," "," "," "],
  [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
  [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
  [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
  [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
  [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
  [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
  [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
  [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
  [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
  [" ","X","X","X"," ","X","X","X","X"," ","X","X","X"," "],
  [" "," "," "," "," "," "," "," "," "," "," "," "," "," "],
  ["X","X","X","X","X","X","X","X","X","X","X","X","X","X"],
  ["X","X","X","X","X","X","X","X","X","X","X","X","X","X"],])

crosswords.append([
  [" "," "," "," "," "," "," "," "," "," "," "," "," "," "],
  [" ","X"," ","X"," ","X","X","X","X"," ","X"," ","X"," "],
  [" ","X"," ","X"," ","X","X","X","X"," ","X"," ","X"," "],
  [" ","X"," ","X"," ","X","X","X","X"," ","X"," ","X"," "],
  [" ","X"," ","X"," ","X","X","X","X"," ","X"," ","X"," "],
  [" ","X"," ","X"," ","X","X","X","X"," ","X"," ","X"," "],
  [" ","X"," ","X"," ","X","X","X","X"," ","X"," ","X"," "],
  [" ","X"," ","X"," ","X","X","X","X"," ","X"," ","X"," "],
  [" ","X"," ","X"," ","X","X","X","X"," ","X"," ","X"," "],
  [" ","X"," ","X"," ","X","X","X","X"," ","X"," ","X"," "],
  [" ","X"," ","X"," ","X","X","X","X"," ","X"," ","X"," "],
  [" ","X"," ","X"," ","X","X","X","X"," ","X"," ","X"," "],
  ["X","X","X","X","X","X","X","X","X","X","X","X","X","X"],
  ["X","X","X","X","X","X","X","X","X","X","X","X","X","X"],])

crosswords.append([
  [" "," "," "," "," "," "," "," "," "," "," "," "," "," "],
  ["X","X","X","X","X","X","X","X","X","X","X","X","X"," "],
  [" "," "," "," "," "," "," "," "," "," "," "," ","X"," "],
  ["X","X","X","X","X"," ","X","X","X","X","X","X","X"," "],
  ["X","X","X","X","X"," ","X","X","X","X","X","X","X"," "],
  ["X","X","X","X","X"," ","X","X","X","X","X","X","X"," "],
  ["X","X","X","X","X"," ","X","X","X","X","X","X","X"," "],
  ["X","X","X"," "," "," "," "," "," "," "," "," "," "," "],
  ["X","X","X","X","X"," ","X","X","X","X","X","X","X"," "],
  ["X","X","X","X","X"," ","X","X","X","X","X","X","X"," "],
  ["X","X","X","X","X"," ","X","X","X","X","X","X","X"," "],
  ["X","X","X","X","X"," ","X","X","X","X","X","X","X"," "],
  ["X","X","X","X","X"," ","X","X","X","X","X","X","X"," "],
```

```
  ["X","X","X","X","X"," ","X","X","X","X","X","X","X"," "],])

crosswords.append([
  [" "," "," "," "," "," "," "," "," "," "," "," "," "," "," "," "," "],
  ["X","X","X","X","X","X","X","X","X","X","X","X","X","X"," "],
  [" "," "," "," "," "," "," "," "," "," "," "," "," "," "," ","X"," "],
  ["X","X","X","X","X","X"," ","X","X","X","X","X","X","X"," "],
  ["X","X","X","X","X","X"," ","X","X","X"," ","X","X","X"," "],
  ["X","X","X","X","X","X"," ","X","X","X"," ","X","X","X"," "],
  ["X","X","X","X","X","X"," ","X","X","X"," ","X","X","X"," "],
  [" "," "," "," "," "," "," "," "," "," "," "," "," "," "," "," "," "],
  ["X","X","X","X","X","X"," ","X","X","X"," ","X","X","X"," "],
  ["X","X","X","X","X","X"," ","X","X","X"," ","X","X","X"," "],
  ["X","X","X","X","X","X"," ","X","X","X"," ","X","X","X"," "],
  ["X","X","X","X","X","X"," ","X","X","X"," ","X","X","X"," "],
  ["X","X","X","X","X","X"," ","X","X","X"," ","X","X","X"," "],
  ["X","X","X","X","X","X"," ","X","X","X"," ","X","X","X"," "],])

crosswords.append([
  [" "," "," "," "," "," "," "," "," "," "," "," "," "," "," "," ","X"],
  ["X","X","X","X","X","X"," ","X","X","X","X","X","X","X"," ","X"],
  [" "," "," "," "," "," "," "," "," "," "," "," "," "," "," "," ","X"],
  ["X","X","X","X","X","X"," ","X","X","X","X","X","X","X"," ","X"],
  ["X","X"," ","X","X","X"," ","X","X","X","X","X","X","X"," ","X"],
  ["X","X"," ","X","X","X"," ","X","X","X","X","X","X","X"," ","X"],
  ["X","X"," ","X","X","X"," ","X","X","X","X","X","X","X"," ","X"],
  ["X"," "," "," "," "," "," "," "," "," "," "," "," "," "," ","X"],
  ["X","X"," ","X","X","X"," ","X","X","X","X","X","X","X"," ","X"],
  ["X","X"," ","X","X","X"," ","X","X","X","X","X","X","X"," ","X"],
  ["X","X"," ","X","X","X"," ","X","X","X","X","X","X","X"," ","X"],
  ["X","X"," ","X","X","X"," ","X","X","X","X","X","X","X"," ","X"],
  ["X","X"," ","X","X","X"," ","X","X","X","X","X","X","X"," ","X"],
  ["X","X"," ","X","X","X"," ","X","X","X","X","X","X","X","X","X"],])
```

3.5 diversite.txt

```
la diversité,diversity,n,f
la mixité ethnique,ethnic variety,n,f
le respect,respect,n,m
se méfier de,to mistrust,v
la religion,religion,n,f
s'entendre avec,to get on with,v
l'acceptation,acceptance,n,f
la citoyenneté,citizenship,n,f
l'égalité,equality,n,f
la peur,fear,n,f
la gentillesse,kindness,n,f
la reconnaissance,recognition,n,f
l'homosexualité,homosexuality,n,f
être ouvert à,to be open to,v
l'élargissement,broadening,n,m
la race,race,n,f
la sexualité,sexuality,n,f
l'exclusion,social exclusion,n,f
tolérer,to tolerate,v
la discrimination,discrimination,n,f
la minorité ethnique,ethnic minority,n,f
multiculturel,multicultural,a,m
multiculturelle,multicultural,a,f
la laïcité,secularism,n,f
la tolérance,tolerance,n,f
respecter,to respect,v
se fier de,to trust,v
la culture,culture,n,f
l'enrichissement,enrichment,n,m
le sectarisme,bigotry,n,m
le citoyen,citizen,n,m
la foi,faith,n,f
l'identité,identity,n,f
la paix,peace,n,f
la coopération,cooperation,n,f
le mythe,myth,n,m
bisexuel,bisexual,a,m
bisexuelle,bisexual,a,f
la nationalité,nationality,n,f
la cohésion sociale,social cohesion,n,f
harceler,to bully,v
```

discriminer,to discriminate,v
la compréhension,understanding,n,f
le genre,gender,n,m
le bénéfice,benefit,n,m
la croyance,belief,n,f
le milieu,background,n,m
égal,equal,a,m
égale,equal,a,f
la différence,difference,n,f
l'ethnicité,ethnicity,n,f
l'inclusion,inclusion,n,f
la richesse,richness,n,f
assimiler,to assimilate,v
l'intégration,integration,n,f
la communauté,community,n,f
stigmatiser,to stigmatise,v
le melting pot,melting pot,n,m
la classe sociale,social class,n,f
hétérogène,heterogeneous,a,m
le handicap,disablement,n,m
adhérer à,to embrace-to adopt,v
semblable,similar,a,m
désapprouver,to disapprove of,v
l'incompréhension,misunderstanding,n,f
la discrimination positive,positive discrimination,n,f
réaliser,to achieve,v
bénéficier de,to benefit from,v
juste,fair,a,m
l'individu,individual,n,m
sensible,sensitive,a,m
la cadre légal,legal framework,n,f
discriminatoire,discriminatory,a,m
enrichir,to enrich,v
intégrer,to integrate,v
éduquer,to educate,v
la ségrégation,segregation,n,f
jouir de,to enjoy,v
la variété,variety,n,f
homogène,homogeneous,a,m
handicapé,disabled,a,m
handicapée,disabled,a,f
l'accès,access,n,m
la valeur,value,n,f

opprimer,to oppress,v
mal interpréter,to misunderstand,v
la poudre aux yeux,tokenism,n,f
la barrière,barrier,n,f
améliorer,to improve,v
accueillant,welcoming,a,m
accueillante,welcoming,a,f
l'ambiance,atmosphere,n,f
améliorer,to improve,v
l'apport,contribution,n,m
l'assimilation,integration,n,f
assurer,to maintain,v
l'attache,link,n,f
l'avis,opinion,n,m
coexister,to live together,v
la colocation,house sharing,n,f
la compétence,skill,n,f
construire,to build,v
contester,to argue,v
le côntrole,control-checking,n,m
décourager,to discourage,v
le défi,challenge,n,m
enrichir,to enrich,v
l'enrichissement,enrichment,n,m
l'énumération,catalogue-list,n,f
équilibré,balanced,a,m
équilibrée,balanced,a,f
l'étendard,standard-flag,n,m
favoriser,to favour-to promote,v
au fil du temps,as time goes by,p
franchir,to break through-to cross,v
indiscutable,unquestionable,a,m
l'individualité,individuality,n,f
inéluctable,inevitable,a,m
insoluble,without a solution,a,m
intégrer,to integrate,v
interagir,to interact,v
se manifester,to show,v
mettre fin à,to put an end to,v
meurtrier,deadly,a,m
meurtrière,deadly,a,f
l'objectif,objective-aim,n,m
paisible,peaceful,a,m

partager,to share,v
la particularité,unique quality,n,f
permettre,to allow,v
la perte,loss,n,f
prévoir,to foresee,v
le processus,process,n,f
la racine,root,n,f
la reconnaissance,recognition,n,f
la richesse,richness,n,f
sain,healthy,a,m
saine,healthy,a,f
sensibiliser,to make aware,v
la singularité,uniqueness,n,f
songer,to think about,v
souhaitable,desirable,a,m
soutenir,to support,v
tenir à,to beleive in,v
la tolérance,tolerance,n,f
valoriser,to value,v

3.6 marginalise.txt

```
pauvre,poor,a,m
la misère,poverty,n,f
exclu,excluded,a,m
exclue,excluded,a,f
appauvrir,to impoverish,v
le chômage,unemployment,n,m
le demandeur d'emploi,job seeker,n,m
toucher une allocation,to receive a benefit,v
l'indifférence,indifference,n,f
apporter des soins,to care for,v
la chaleur humaine,human warmth,n,f
vivre dans la misère,to live in poverty,v
la santé,health,n,f
être dans le besoin,to be in need,v
la souffrance,suffering,n,f
caritatif,charitable,a,m
caritatif,charitable,a,f
la faim,hunger,n,f
le ghetto,ghetto,n,m
l'appauvrissement,impoverishment,n,m
le système sanitaire,health system,n,m
défendre,to defend,v
mendier,to beg,v
le toxicomane,drug addict,n,m
la pauvreté,poverty,n,f
marginaliser,to marginalise,v
l'exclusion sociale,social exclusion,n,f
l'emploi,employment,n,m
le chômeur,unemployed person,n,m
le sans-abri,homeless person,n,m
le soin,care,n,m
le centre de distribution alimentaire,food bank,n,m
le droit,right,n,m
le seuil de pauvreté,poverty threshold,n,m
le besoin fondamental,basic need,n,m
souffrir,to suffer,v
l'association,charity,n,f
affamé,starving,a,f
affamé,starving,a,f
la zone sensible,difficult area,n,f
démuni,destitute,a,m
```

démunie,destitute,a,m
aisé,well off,a,m
aisée,well off,a,m
le marginalisé,excluded person,n,m
l'accompagnement juridique,legal aid,n,m
le mendiant,beggar,n,m
se droguer,to take drugs,v
le bénévole,unpaid volunteer,n,m
le bénévolat,volunteering,n,m
la frange,fringe,n,f
la retraite,retirement,n,f
retraité,retired,a,m
retraitée,retired,a,f
le mode de vie,way of life,n,m
le nomadisme,wandering about,n,m
le soutien,support,n,m
la générosité,generosity,n,f
le sanctuaire,sanctuary,n,m
à l'étranger,abroad,n,m
autrui,other people,n,m
interpeller,to arrest,v
assistant social,social worker,n,m
donner le coup de main,to lend a helping hand,v
la réinsertion,reintegration,n,f
en détresse,in distress,n,m
le foyer,hostel,n,m
l'autorité locale,local authority,n,f
s'occuper de,to look after,v
le volontaire,paid volunteer,n,m
survivre,to survive,v
le structure d'accueil,reception centre,n,m
le troisième âge,old age,n,m
prendre la retraite,to retire,v
errer,to wander,v
distribuer,to hand out,v
soutenir,to support,v
donner de son temps,to give your time,v
le refuge,shelter,n,m
le quartier défavorisé,poor neighbourhood,n,m
héberger,to put up shelter,v
défavorisé,underprivileged,a,m
défavorisée,underprivileged,a,f
le SDF,homeless person,n,m

le service social,social services,n,m
l'aide à domicile,home help,n,m
la détresse,distress,n,f
apporter de l'aide,to give help,v
faire un service,to do a favour,v
sensibiliser,to raise awareness,v
priviliégié,privileged,a,m
priviliégiée,privileged,a,f
les pouvoirs publics,authories,n,m
donner à manger,to feed,v

3.7 criminels.txt

```
la prison,prison,n,f
un détenu,prisoner-inmate,n,m
un prisonnier,prisoner,n,m
en tôle,in jail,n,m
traiter,to treat,v
l'incarcération,imprisonment,n,f
emprisonner,to imprison,v
l'emprisonnement,imprisonment,n,m
la réclusion à perpétuité,life imprisonment,n,f
derrière les barreaux,behind bars,n,m
la peine de mort,death penalty,n,f
la peine capitale,death penalty,n,f
une peine,sentence,n,f
protéger,to protect,v
dissuader,to deter,v
punir,to punish,v
sanctionner,to punish,v
la punition,punishment,n,m
les travaux d'intérêt général,community work,n,m
la réinsertion,rehabilitaion,n,f
la réhabilitation,rehabilitaion,n,f
la réintégration,reintegration,n,f
aller en prison,to go to prison,v
la liberté,freedom,n,f
la liberté conditionnelle,parole,n,f
libérer sur parole,to parole,v
une évasion,escape,n,f
incarcérer,to imprison,v
un évadé,escapee,n,m
une cellule,cell,n,f
un gardien de prison,prison guard,n,m
la courm,exercise yard,n,f
les stupéfiants,drugs,n,m
se droguer,to take drugs,v
un drone,drone,n,f
introduire clandestinement,to smuggle,v
partager une cellule,to share a cell,v
l'isolement,solitary confinement,n,m
une prison de haute sécurité,high security prison,n,f
le taux d'incarcération,imprisonment rate,n,m
une prison ouverte,open prison,n,f
```

le harcèlement,bullying,n,m
la violence,violence,n,f
le surpeuplement,overcrowding,n,m
la dépression,depression,n,f
la santé mentale,mental health,n,f
la dépendance,addiction,n,f
l'arrestation,arrest,n,f
le comportement,behaviour,n,m
brutal,brutal,a,m
brutale,brutal,a,f
confisquer,to confiscate,v
une bagarre,fight,n,f
un meurtre,murder,n,m
une agression,attack,n,f
un gang,gang,n,m
la libération précoce,early release,n,f
un cambriolage,burglary,n,m
la loi,law,n,f
un délinquant,offender,n,m
la sécurité,safety,n,f
la justice,justice,n,f
le rêglement,rules,n,m
la victime,victim,n,f
la politique,policy,n,f
scanner,to scan,v
les murs,walls,n,m
le budget,budget,n,m
l'investissement,investment,n,m
le personnel,staff,n,m
le compagnon,bunk beds,n,m
l'activité,activity,n,f
l'autorité,authority,n,f
une raclée,beating,n,f
l'ennui,boredom,n,m
l'inactivité,inactivity,n,f
un visiteur,visitor,n,m
le manqué de personnel,lack of staff,n,m
la vétusté,old age,n,f
privatiser,to privatise,v
privé,private,a,m
privée,private,a,f
la maladie mentale,mental illness,n,f
la discipline,discipline,n,f

l’infirmerie,infirmary,n,f
récidiver,to reoffend,v
un multirécividiste,habitual offender,n,m
s’ennuyer,to get bored,v
le suicide,suicide,n,m
se suicider,to commit suicide,v
faire de la prison,to serve time,v
un séjour en prison,prison stretch,n,m
l’opinion,public opinion,n,f
une inspection,inspection,n,f
la fouille,search,n,f
cacher,to hide,v
dissimuler,to hide,v
fouiller,to search,v

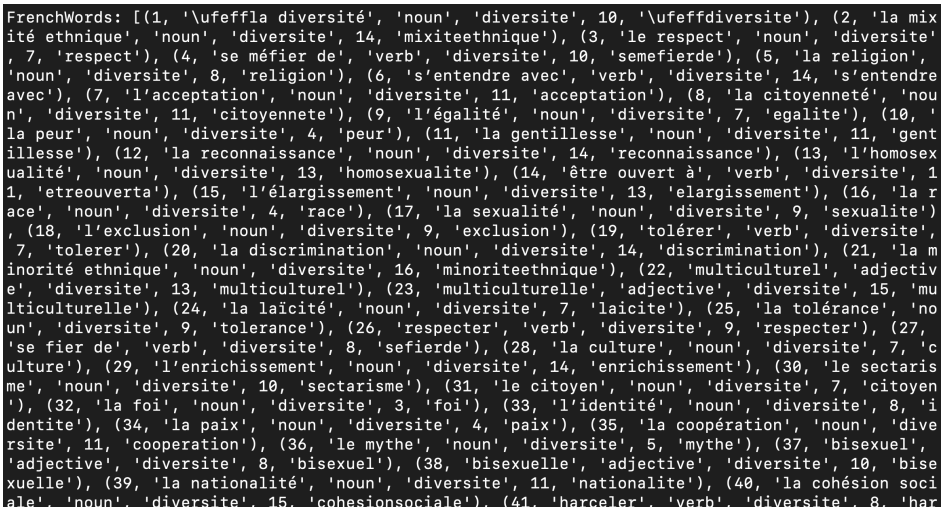
4 Testing

Screenshot Reference	Purpose of Test	Data Used	Expected Outcome	Results
Figure 11	This was to see whether the database-compiler.py program would correctly convert the ".txt" vocabulary files to a database using SQL UPDATE queries (for objective 1.2).	The database-compiler.py program was ran with the three text files shown in Section 3 being inputted into it.	The program should print onto the screen a representation of the database it has constructed using the text files. The database should be in a correctly normalised format containing all of the data from the text files.	The output shows the contents of three of the database tables displayed in a list format where each list within the list contains a record from the table. The output is as expected: every word is featured in the table, has a unique key and is correctly linked to its translation in the Translations table.
Figure 12a	This was to see whether the crossword-generator.py algorithm for converting the crossword grid to a graph worked (for objective 3).	The crossword grid used is the one that can be seen in the *crossword-grids.py* program on line 33.	The algorithm should correctly pick out the word lines from the crossword grid and assign nodes to represent them. The points of intersection for each node should be correctly displayed.	The output correctly shows the 4 nodes, their intersections and locations on the grid. The first 4 outputs show the intersections in the format "node: node-it-intersects-with [point of intersection]".

Figure 12b, 12c	This was to see whether the crossword-generator.py traversal algorithm followed the correct procedures (for objective 3.2).	The crossword used for this traversal is the one that can be seen in the *crossword-grids.py* program on line 33. The vocabulary used was from *diversite.txt* and *criminalite.txt* and the filling in language was French.	The algorithm should correctly perform a depth-first traversal where it tries to attribute each node an appropriate word and works backwards if no suitable words can be found.	Firstly, all of the query results seem appropriate. The process it follows also seems correct: it starts from node 1, performs a query for it, chooses a random word from that query, traverses to the next node (node 2), does the same for that node, traverses to the node it intersects with (4) and does the same. At this point it traverses to node 3 which does not yield any query results. Because of this it has to return to node 4 and try a new word.

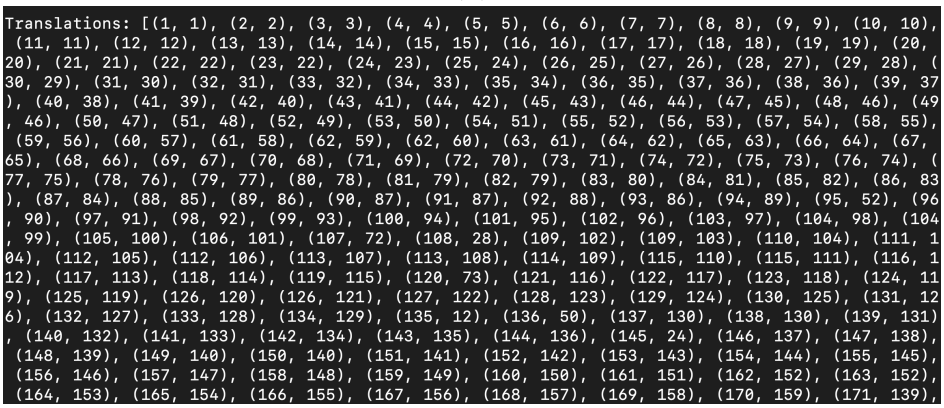
Figure 13	This test was to see whether the GUI worked as intended (for objectives 2 and 4).	The user should be able to select their specifications, be presented with an appropriate crossword grid and then be able to fill it in and check their answers.	In Figure 13a you can see the user has been able to toggle three buttons to indicate that they want an 8x8 crossword grid composing of unit 1 vocabulary and that they want to fill the grid in using French (with English clues). Figure 13b shows that a grid of the correct size has been created with appropriate vocabulary. Figure 13c shows that letters can be entered into the grid and figure 13d shows that the "check crossword" button can be pressed to reveal a score and the answers.
Figure 14	This test was to see whether different specifications for the crossword would also work.	You can see that this time a crossword grid of size 12x12 has been produced with the clues in French.	

Figure 15	The aim of this test was to put the algorithm to its limit by selecting only one vocabulary unit and selecting the 14x14 grid size which is the hardest to construct.	Knowing the program's limitations (discussed in section 2.3.4), it will likely fail to construct an appropriate crossword.	What you can see in Figure 15b is the fail state which happens when the algorithm fails to find any way to construct a crossword given the specifications.



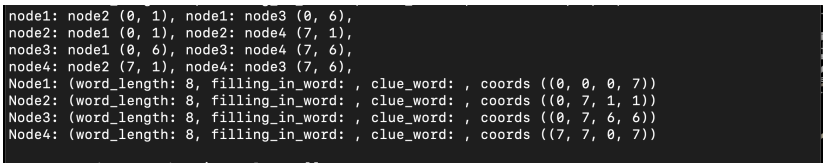
(a)

(b)

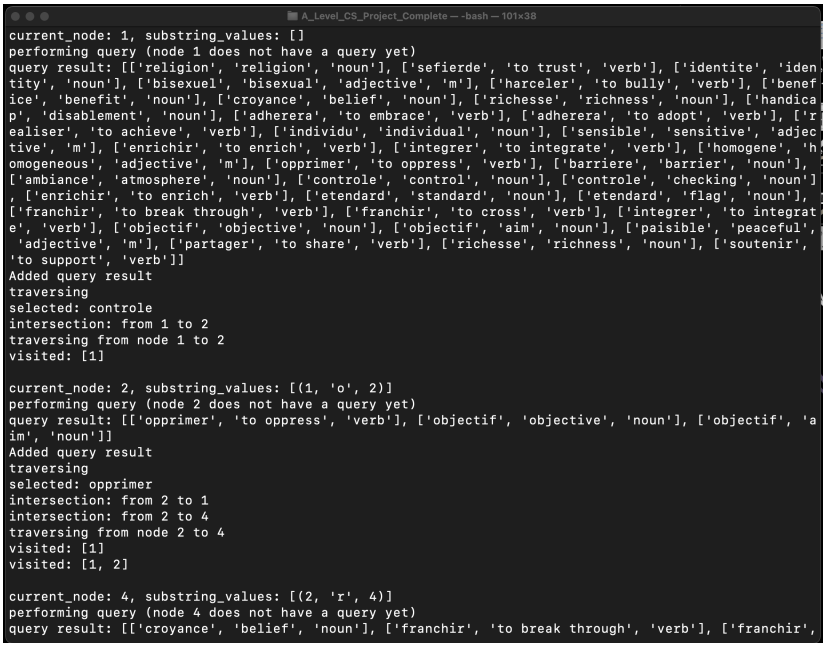


(c)

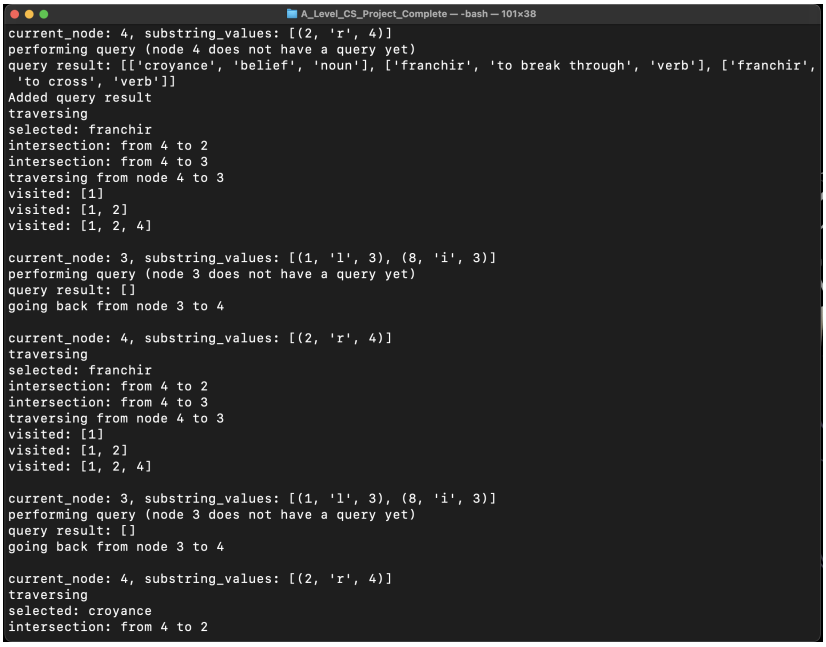
Figure 11: Screenshots of the outputs to the command line when the database-compiler.py was ran.



(a)

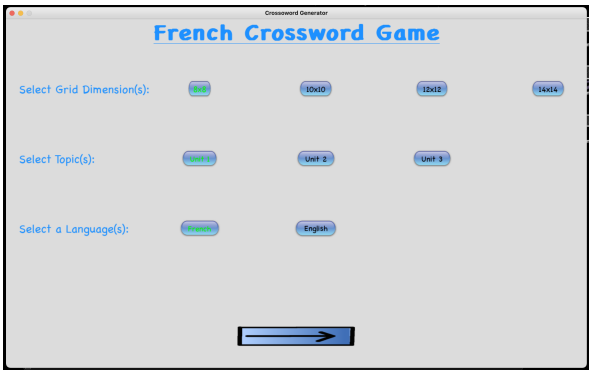


(b)

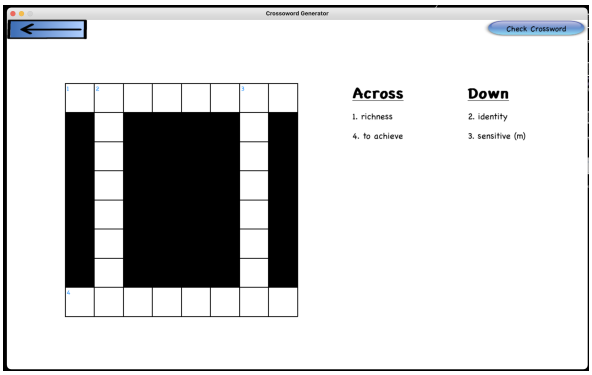


(c)

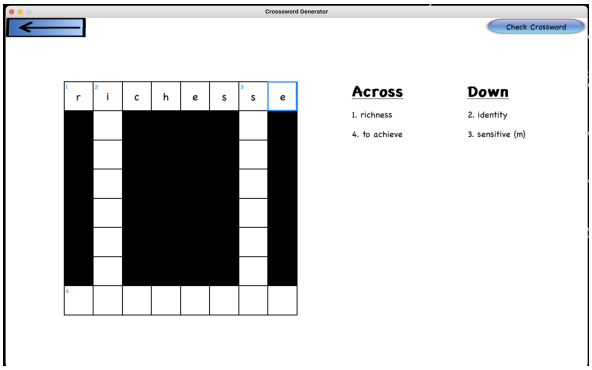
Figure 12: Screenshots which shows the outputs of the crossword-generator.py program when debugging is set to True.



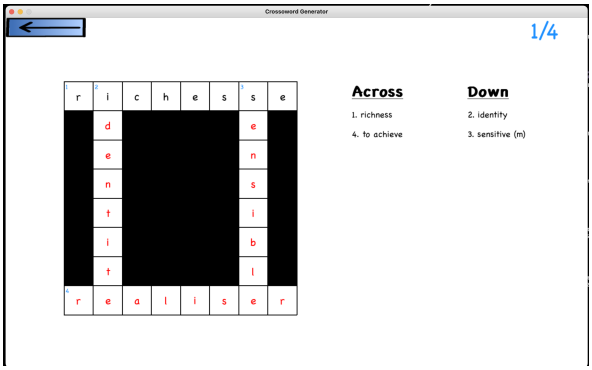
(a)



(b)

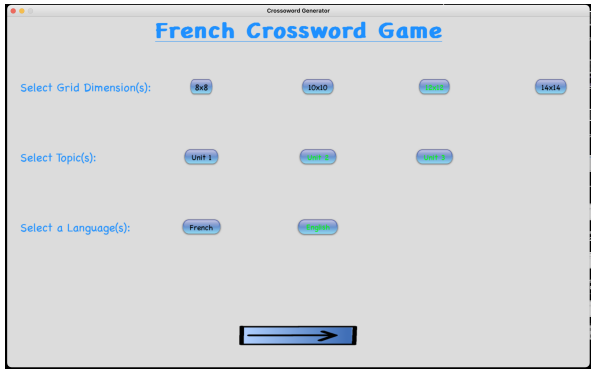


(c)

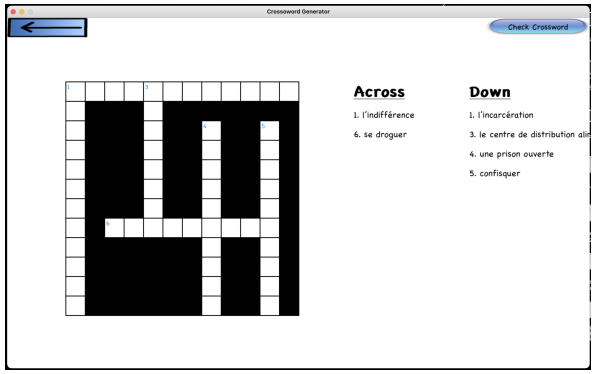


(d)

Figure 13: Screenshots to show the process of selecting specifications from the menu, filling in the crossword and checking your answers.

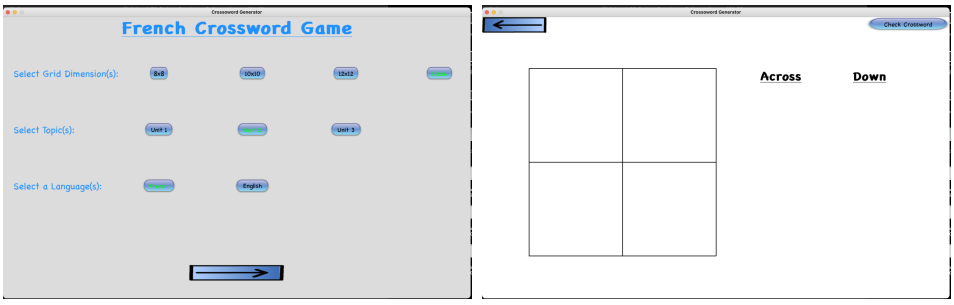


(a)



(b)

Figure 14: Screenshots to shows how different specifications will result in a completely different crossword.



(a) (b)

Figure 15: Screenshots to show how the algorithm can fail to construct an appropriate crossword.

5 Evaluation

5.1 Completion of Objectives

The following table shows where the objectives made in section 1.6 have been completed amongst the Python files:

Objective	File	Location	Notes
1.1, 1.1.1	crossword-grids.py		This file provides the crossword generator.py file with a list of empty crossword grids. There are grids ranging from size 8x8 to size 14x14. There are at least 6 differently arranged grids for each grid size.
1.2, 1.2.1, 1.2.2	database.db, database-compiler.py		The database-compiler.py file adds all of the words from the .txt files to the database.db file using SQL. The program makes sure not to include duplicates and the tables conform the third normal form.
2.1.1, 2.1.2, 2.1.3, 2.3	main.py	MenuScreen(), line 562	There are *ButtonSets* which the user can toggle to indicate their specifications. The window exit button allows the user to quit the program at any time (this is regulated using the *program-loop* attribute).
2.2	main.py	CrosswordScreen(), line 657	The *back-button* allows the user to return the the *MenuScreen()*.

3.1	crossword-generator.py	main(), line 566	The crossword list is imported from crossword-grids.py, shuffled and a random one is chosen for the traversal.
3.2	crossword-generator.py	Crossword.query(), line 298	A SQL query is formed based on the users specifications which makes sure only the appropriate words are used to fill in the crossword. It also checks to see that none of the words have already been placed into the crossword.
3.3	crossword-generator.py	WordLine(), line 17	Every node is allocated a unique number to identify it. This number decides which order to traverse in.
4.1	main.py	Crossword(), line 191	Depending on the grid size a certain number of lines are drawn in an intersecting pattern to represent the grid. Blank grid squares are filled in black.
4.2	main.py	Crossword(), line 244	The first grid square of each word line is allocated a number which is displayed in blue in the top left corner. If there is both a word going across and down from that point then that is noted.

4.3	main.py	ClueList(), line 33	This class is used to display a numbered list of the clue words. There is one column for across and another for down.
4.4	main.py	GridSquare(), line 78	When the user clicks their mouse *mouseHovering()* checks if they are hovering over a grid square and if they are, that grid square becomes activated meaning any letter entered will be placed into that square.
4.4.1	main.py	CrosswordScreen(), line 735	When the user enters a key, the program checks if it has the unicode value of a lower-case letter so there's no way the user can enter an inappropriate character.
4.5, 4.6	main.py	CrosswordScreen(), line 668	The user can press the *check-button* (a one-time use button) to display their score and show them where they went wrong. There is no button to ask for help because I decided it was unnecessary: if the user wants to see the answers, they can just use the *check-button*.

5.2 Effectiveness of Solution

As can be seen in section 5.1, every major objective has been met by my solution. A wide range of personalised crosswords can be generated which allow the user to engage in useful revision of vocabulary. The interface is simple but effective and provides an

appropriate level of interactivity.

I think there could still be more options available to the user and there could be more opportunities for feedback. The limitations discussed in Section 2.3.4 result in a limit to the maximum possible size and complexity of crosswords but this does not inhibit its educational potential. The crosswords it generates are still challenging and varied. The limitations also mean that some of the specifications selected by the user can result in a crossword failing to be formed. This would ideally need to be improved in some way.

5.3 Comments from End-User

In showing the project to my end-user as well as some other test users, I was able to receive some feedback. Firstly, no-one had any difficulty understanding how to use the program and it was generally agreed that the interface was intuitive and effective. They all felt as though the algorithm was able to generate a range of different crosswords tailor fitted to their specifications.

One person commented that the program could do with having more varied clues which were not simply asking the user to translate each word. For example, it could give a cryptic clue which you would have to decipher. Another suggested that there could be a button to ask for some help with some of the crossword answers if the player ever felt stuck.

Some made the point that the vocabulary available only covered half of the French A level specification and so more would need to be added. A few also noted that the complexity of the crossword grid patterns could have been higher to add more of a challenge. That being said, the vast majority found the crossword proficiently challenging and were not able in every circumstance to fully complete the crossword. Part of the reason for this is that for any given french word there are many possible translations and finding the specific word needed for the crossword can sometimes be difficult.

My end-user was interested in the idea of creating a system whereby students could be given the game to play in their own time and teachers would be able to see their students' activity. In relation to this, she thought it would be useful to try and implement my extension objective number 7.

5.4 How the Solution Can Be Improved

Taking this forward, I would certainly try to include more vocabulary, grid size options and grid complexity. In order to tackle the issues raised by the use of recursion, I could switch to using other loop methods such as using while loops. Including more vocabulary would also improve the algorithm's likelihood of finding a solution. At the moment, there is a chance the user can select menu options which result in no solution being found. I can tackle this by forcing the user to select an adequate number of vocabulary units (especially if they are selecting one of the more complex grid sizes).

Achieving my extension objectives would be another great way to improve the program. In order to construct more complex crossword clues, I would likely need to add a lot more fields to the database in order to store useful information about each word. Similarly, I would need to store multiple details after each crossword to give more feedback to the user and to construct more tailor fitted crosswords.

The teacher feedback idea would be an important step forward if I wanted to improve the usability of this game in education. Doing this would likely involve networking and account management.

Transcript 1

Me:	If you were gonna have someone make a French learning game for you (.) what kind of features would you like (?)
Ms Roddy:	Would it be for me as a teacher (?)
Me:	I mean (.) for students to use
Ms Roddy:	From a teacher's perspective (.) I think it would be good to have something erm where like you can track their progress (.) you know so you can see how long they've spent on it or how many points they've got or something (.) do you mind if I ask these two (?)
Me:	Uh no I don't mind
Ms Roddy:	Okay guys if you had a new app for learning French (.) what features would be good to have (?)
Y10 boy 1:	Listening
Ms Roddy:	Yeh because there's not many apps actually that do good listening so listening would be really good to have (2) maybe even something with speaking where they have the opportunity to I don't know record themselves and listen to it
Me:	So if it was gonna be something where it was asking questions (.) what kind of questions would it be best to ask (?) like grammar or
Ms Roddy:	= I think there's lots of stuff out there for grammar but what it would be good to have is more context but either where it's linked more closely to a textbook because at the moment all the apps out there are a bit random (.) whereas if it was linked a bit more closely to a textbook then that would be more relevant

Transcript 2

Me:	So do you find your students have difficulty learning vocabulary (?)
Ms Roddy:	Yeh so I think one of the problems you have as a teacher is that people can often be quite good at learning for a test so they'll learn it for Monday's test but then a week later they've forgotten it when they've seen it in context (.) so I suppose in terms of your app it would be good to have um kind of the facilities to go back to older vocab and have that running through again (.) it's called retrieval is the word they use in language teaching
Me:	So what mistakes to students make when they try to revise for vocab tests
Ms Roddy:	I suppose the most common mistakes in French for example are just spellings and missing accents off (.) and like I said I think the biggest issue is then they forget it
Me:	Do you think video games would be useful for education like generally in education do you think they are helpful (?)
Ms Roddy:	Yeh I think they've got a lot of potential I think things like Blooket and Kahoot obviously people really enjoy them (.) um one thing that I think it's really important to be aware of is making sure it's still education obviously it's still got to be enjoyable but still educational enough (.) so like have you seen Blooket (?)
Me:	= I don't think so

Ms Roddy:	= We've not used it in class it's like quite a new combination between Quizlet and Kahoot so it's more vocabulary based but you do the picking between four options but the format it uses isn't very good because the answers are always obvious so even though it's good game it isn't very useful
Me:	So more specifically I have planned for my project rather than making a big game with loads of different modes and stuff it's just going to be a program that generates loads of random crosswords for French (.) do you think that would be effective (?)
Ms Roddy:	How would it work when you say random crosswords so would it be how I as a teacher could say here's a crossword on like sports vocabulary (.) would I create the words myself (?)
Me:	So it's going to have a database of French words um I'm planning on it just being A level split into topics and so you as the user will select the topics you want to do (.) size of the crossword grid and also which language you want to fill it in with
Ms Roddy:	I think that's a brilliant idea
Me:	Is there anything specific you'd want from the user interface
Ms Roddy:	Will the questions just be like "education" and then you have to write "éducation"
Me:	I'm planning simply on just the (.) like the hints on the side (.) being simple translations (.) but if I were going to make it more complicated what kind of stuff would you want (?)
Ms Roddy:	I guess you could do more the clues like we do in Articulate so say if you wanted the "president of France" and obviously you'd have to answer Emanuel Macron or something like (.) say you've got the "marginalisé" then you'd put something like "les personnes ont des difficultés en la société" (.) the interface (.) do you mean in terms of what it looks like (?)
Me:	So if it makes a crossword (.) would you want it to be just like a grid and then you type a letter into each square of the grid (.) would you want it to check the answers as you type it in (.) or for there to be a button to press after you've typed everything in to check (?)
Ms Roddy:	I don't know really
Me:	The issue is that as you type it in (.) you could do trial and error
Ms Roddy:	Yeh and then it's not (.) it's a bit bitty rather than seing the full word written down (.) so yeh probably that it checks it afterwards
Me:	And one of my ideas to take it further would be is to ask the user to find synonyms for words (.) so do you think that would be quite good (?)
Ms Roddy:	Yeh I think that's a really good idea
Me:	Do you think it would be good to have the crossword factor in your progress as in (.) you know so if you fill in the crossword and get it wrong for certain words would it be good if it could recognise which words you are struggling with and then in future crosswords be more likely to ask you
Ms Roddy:	Yeh (.) I feel like (.) does Quizlet do that (?)
Me:	= Yeh
Ms Roddy:	= I think that's a really good idea

References

[1] *The COVID-19 pandemic has changed education forever. this is how.* World Economic Forum, 2020, https://www.weforum.org/agenda/2020/04/coronavirus-education-global-covid19-online-digital-learning/.

[2] *Crosswords.* theguardian.com, 2021, https://www.theguardian.com/crosswords/series/quick.

[3] *Home page.* Duolingo.com, 2021, https://www.duolingo.com.

[4] L. A. Annetta, *Video games in education: Why they should be used and how they are being used*, Theory into practice, 47 (2008), pp. 229–239.

[5] R. Gafni, D. B. Achituv, and G. J. Rachmani, *Learning foreign languages using mobile applications*, Journal of Information Technology Education: Research, 16 (2017), pp. 301–317.

[6] G. Kiryakova, N. Angelova, and L. Yordanova, *Gamification in education*, Proceedings of 9th International Balkan Education and Science Conference, 2014.

[7] No-author, *Pygame documentation.* Pygame, 2017, https://www.pygame.org/docs/.

[8] No-author, *Français.* Languages Online, 2021, https://www.languagesonline.org.uk/Hotpotatoes/frenchindex.html.

[9] No-author, *Home page.* Quizlet, 2021, https://quizlet.com/en-gb.

[10] No-author, *Home page.* sqlite, 2021, https://www.sqlite.org/index.html.