

Learn'English

Desktop app for the English module project

Pierre-Arnaud BLANC

Guillaume GARCIA

May 4, 2017



UNIVERSITÉ
DE LORRAINE

Contents

Introduction	2
User documentation	3
Setting up a working session	3
Vocabulary training	6
Check your irregular verbs	7
End of a session	8
Technical documentation	9
Conception	9
Difficultés rencontrées	12
Conclusion	13
Appendix	14
Bibliography	14
Download links	14

Introduction

This report presents our application *Learn'English* as part of the TELECOM Nancy English module. The application aims to help the user into learning specific vocabulary, and most of irregular verbs.

We noticed that the basics, especially irregular verbs and vocabulary, are lacking to most of people who learn English. And as everybody knows, a strong basis is required to improve your level. Therefore, we had the idea of developing a software tool which could make the checking of those bases more playful and convenient for students. Also, in an ergonomic concern, we wanted our application to be the more portable as possible.

Consequently, we designed two main features in our app. The first one, named *Train your vocabulary*, is a translation exercise in order to make you memorize your own vocabulary sheet. We will talk about those custom training sheets later in this report. The second feature, named *Check your irregular verbs*, allows you to train yourself on more than 150 irregular verbs. You can also custom this session to make a more personal workout.

The report will first set forth the *User documentation*, explaining precisely how to use correctly the app. Then, it will pose a *Technical presentation*, written in French, where the exact specification of our app is exposed. Finally, after the conclusion, you will be able to see our bibliography and a link to the executable file (binary) and the source code of the app (*GitHub* link).

User documentation

Learn'English was designed for training the user on the basics about English language. Thus, two main separate features were built around two simple exercises. These exercises include a vocabulary test, which consists of assembly-line translations of words, and an irregular verbs test, where you are asked to give past forms of different irregular verbs. Moreover, you can load specific training sheets for both exercises, and also store your high scores and compare them to those of other students. Therefore, our software become a real and powerful tool to help students to check their basic knowledge. To ensure that, it uses light databases in CSV format.

Setting up a working session

When you launch the app you reach the home window (see **Picture 1**). Then the app will ask you a username, type the one you want, and click the *Log in* button. It will associate your username to your future high scores and load ancient ones.

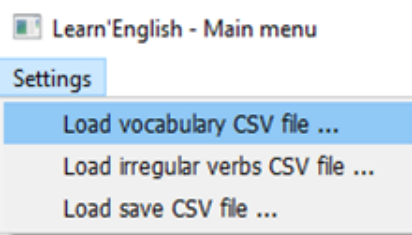
Picture 1: Home window



Note that if there isn't a file named *saves.csv* (the default save file loaded after logging) in the same folder as the executable file *learnEnglish.exe* on Windows, or *learEnglish* on Linux, you'll get an error telling you that the could load the save file. If it happens, do not worry and follow these steps:

- Click on *Settings* in the menu bar.
- Choose *Load save CSV file ...*
- Indicate the path (absolute or relative) to a valid save file, in CSV format, then press *OK*.

As long as no valid and existing save file is specified, you are not permitted to launch any exercise. This problem of non-existing files could also happen once you launch one of the exercises : the necessary databases are missing. You can solve this problem by a similar way (see **Picture 2**).



Picture 2: Settings in the menu bar

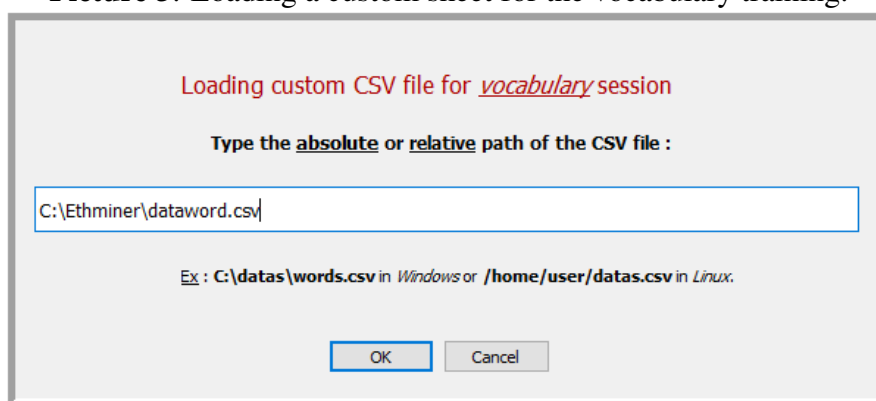
Since you have downloaded normally the archive containing the app (see the download link in the appendix), you should not get these problems because it provides a save file and a database for each exercise. But your may want to load you own sheets of vocabulary or irregular verbs; that's the point of our tool ! To do that, follow these few steps (see **Picture 2**):

- Click on *Settings* in the menu bar.
- Choose *Load vocabulary CSV file ...* or *Load irregular verbs CSV file ...*.

A small window will pop up (see **Picture 3**: example of loading a file for vocabulary training).

- Indicate the path to an CSV file corresponding to the desired exercise, then press *OK*.

Picture 3: Loading a custom sheet for the vocabulary training.



Once you are logged in and all the settings are correctly set up, two start-buttons appear: you are ready to launch an exercise. If your username already exists in the save file, the app will load your own high scores for both exercises and display them above the two buttons (see **Picture 4**). To start a training session, just click on the corresponding button. Note that if you increase your highscore in

an exercise, and you want to update it on the home window, just push the button refresh (the one with the icon).

Picture 4: Logged in and ready to go!



To close the app, just press the *Save and Quit* button. It will automatically store your highscores in the CSV save file (see **Picture 4**). To know more about CSV files, refer to the **Technical documentation**.

Be aware that if you have not configured correctly the app you will not be able to launch any exercise since they use external files (CSV format) to generate words and irregular verbs.

Vocabulary training

At this step, you should have already configured correctly the databases. If not, an error message might appear once you click on the *I'm ready !* button, then you should read the solution in the previous section.

Once the exercise is launched, the app will ask you to translate a given English word into its main meaning in French. Type it in the line-edit, then push the *Validate* button or press Enter key. If it's correct you'll see a *Right !* just below the line-edit. If not, a *Wrong ...* will appear and the app will print the solution (see **Picture 5**).

For each correct answer, you earn one point. You don't earn any point for a wrong one. Your score is displayed and updated all the time in the right corner of the window. A progress bar indicates you how many words are left. When you have validated -or not a word, just click the *Next !* button, another word will be picked up randomly. Words that have already been checked are not showed anymore until the end of the session: a given word appears just once during a session.

Picture 5: Vocabulary exercise sample

The screenshot shows a web application window titled "Verb training". At the top, there are two circular flags: the Union Jack on the left and the French flag on the right. Between them is the text "Train your vocabulary !" in red. Below this, it says "Logged as : AOS". The main question is "What's the french for : water". Below the question is a text input field containing "wat" and a "Validate" button. Below the input field, it says "Wrong ...". At the bottom left, it shows "Your answer : wat" and "Expected answer : eau". At the bottom right, there is a "Progression" bar at 60% and a "Score" display showing the number 2. A "Next !" button is located at the bottom center.

The provided database for vocabulary *datawords.csv* contains almost **40** couples of words. This is a light sheet, but the user can enlarge this database up to 2000 couples of words (limit fixed by the compiler in our code). Once you finish the exercise, a new screen pops up. Refer to the **End of session** section for more info about that.

Check your irregular verbs

As in the **Vocabulary training** section, just push the *I'm ready !* button. If an error message is printed, refer to the solution in the **Setting up a working session** section.

The process here is similar to the vocabulary exercise. The app shows an irregular verb in its infinitive form. Type its preterit form in the first line-edit and the past participle form in the second one, then push *Validate* button or press Enter key. If a form is correct you'll see a *Right !* just below the corresponding line-edit. If not, a *Wrong ...* will appear and the app will print the solution below the wrong answer. The French translation of the verb is also displayed, just for information, on the right frame (see **Picture 6**).

For each correct answer, you earn one point. You don't earn any point for a wrong one; thus you may earn up to 2 points per irregular verb. Your score is displayed and updated all the time in the right corner of the window. A progress bar indicates you how many verbs are left. When you have validated -or not a pair of forms, just click the *Next !* button, another irregular verb will be picked up randomly. Verbs that have already been checked are not showed anymore until the end of the session: a given verb appears just once during a session.

Picture 6: Irregular verbs exercise sample

The screenshot shows a window titled "Verb training". At the top, there are two circular flags: the Union Jack on the left and the French flag on the right. The main title "Check your irregular verbs !" is centered in red. Below it, "Logged as : AOS" is displayed. The instruction "Fill the blanks (+1 point for each if correct) :" is in red. The exercise is divided into four columns: "Infinitive form" with the word "arise", "Preterite form" with a text input containing "arose" and a green "Right !" message below it, "Past participle form" with a text input containing "don't know" and a red "Wrong ..." message below it, and "French meaning" with the word "survenir". Below the "Preterite form" input, the "Correct answer : arisen" is shown. A "Next !" button is centered below the inputs. On the right side, there is a "Progression" bar at 80% and a "Score" box displaying the number "6".

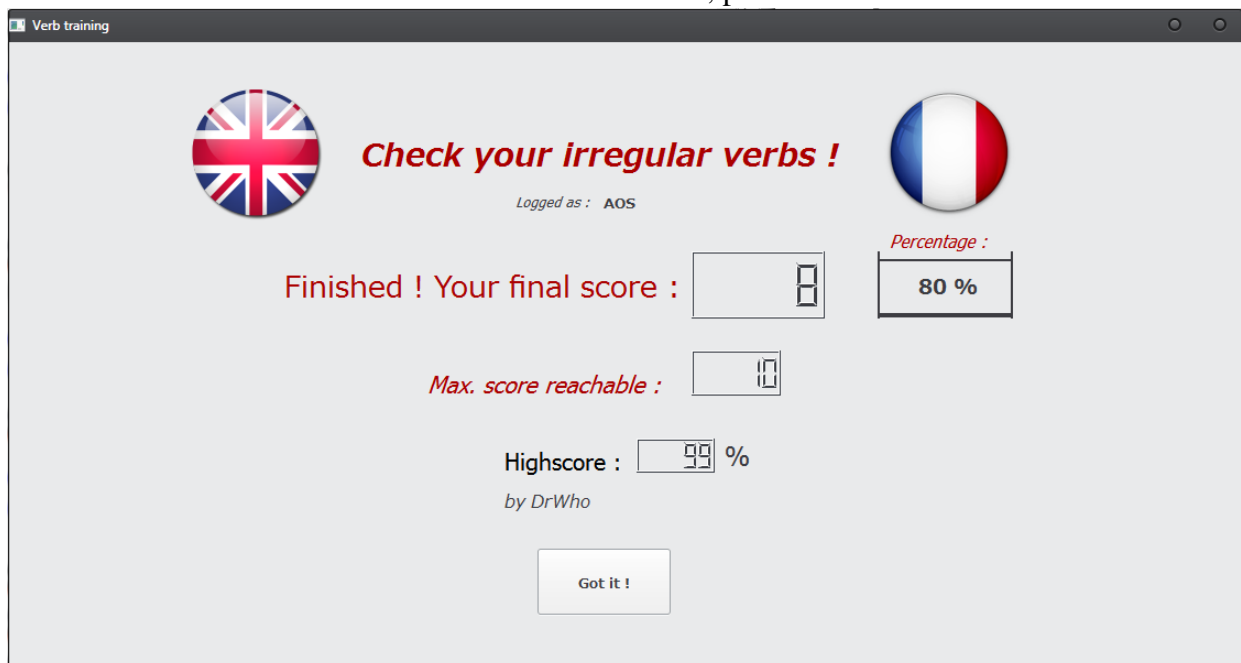
The provided database for irregular verbs *dataverb.csv* contains exactly **152** irregular verbs. Though this is a large sheet, the user can enlarge this database up to 2000 irregular verbs with its three forms plus the French translation (limit fixed by the compiler in our code). Despite this limit, there are about 200 irregular verbs in English in normal use... Once you finish the exercise, a new screen pops up. Refer to the **End of session** section for more info about that.

End of a session

When you finish an exercise, a special ending screen is displayed (see **Picture 7**). It's almost the same for both training. It shows your final score, and your percentage calculated by the ratio of your final score to the maximum score reachable (which is also printed). The highest score recorded in the save file for this exercise is also displayed, and below it you will see the username of the best player associated to this highscore.

Once you're done, just click on the *Got it !* button. This will close this training and you'll reach the home windows. To update your new highscore on the screen, just press the *Refresh* button (the one with the arrow) in the home window. Once an exercise is done, you cannot re-launch it until you quit the app, by clicking the *Save and Quit* button (see **Picture 4**). It saves your highscores only if they are better than the ancien ones.

Picture 7: End of session, print scores



Technical documentation

Conception

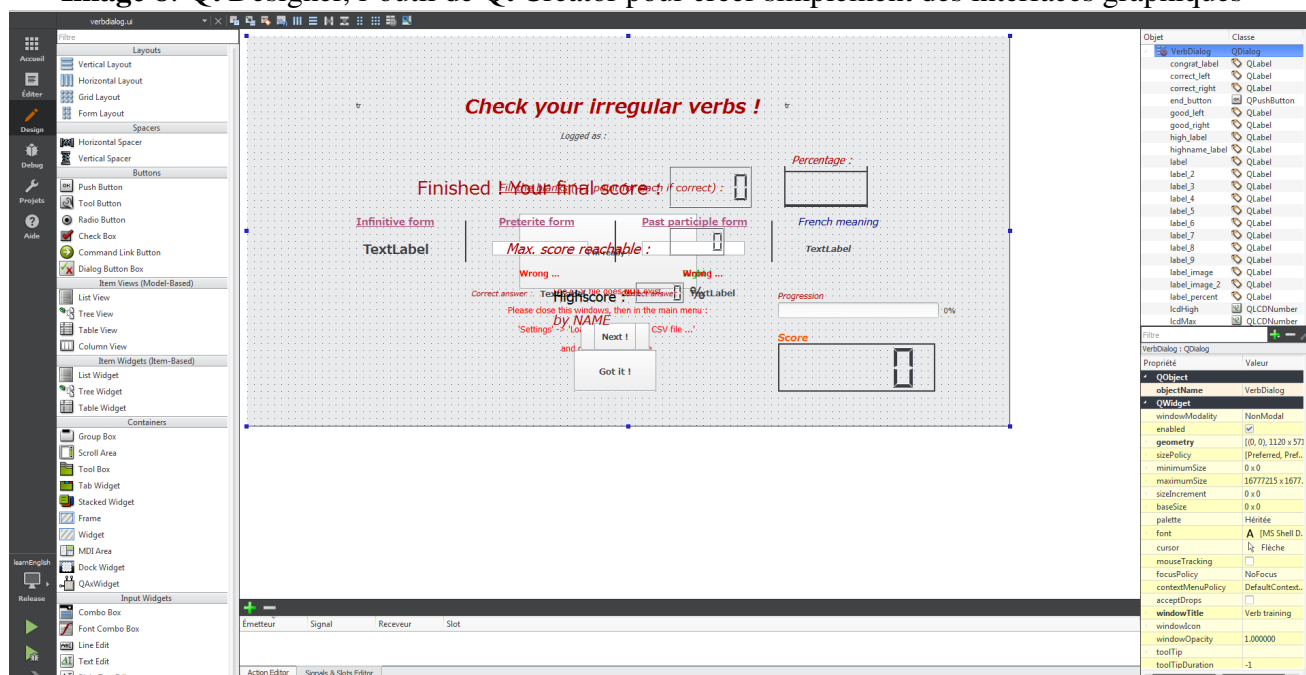
Le choix de conception étant totalement libre, nous avons d'entrée de jeu choisi de sortir des sentiers battus. Un des langages les plus utilisés aujourd'hui dans les entreprises de software est le C++, c'est aussi un langage très répandu dans le *Logiciel Embarqué*, ce qui correspond à notre approfondissement (LE). Nous n'avons donc pas hésité à choisir le C++, afin de nous former dans ce langage. Une telle formation est un atout non négligeable sur un C.V. de jeune ingénieur spécialisé en logiciels embarqués.

Bien évidemment, concevoir une application bureau, donc graphique, est une tâche bien plus ardue que le développement d'un simple programme en ligne de commande. Les bibliothèques standards *GNU/Linux* étant relativement pauvres et peut satisfaisantes pour notre objectif, nous avons décidé d'utiliser un **framework**, c'est-à-dire un cadre applicatif ou une infrastructure de développement, bien connu dans le domaine industriel et relativement bien répandu. Ce framework s'appelle *Qt*. Nous l'avons utilisé dans sa version 5.8 (dernière version stable en date). Ce framework est spécialisé dans le développement d'application bureau. De plus, *Qt* est disponible aussi bien sur *Linux* que sur *Windows* (nous avons développé notre application sur les 2 systèmes d'exploitations). Il était ainsi beaucoup confortable de concevoir notre application.

L'autre avantage de *Qt* est la portabilité des applications construites avec. En effet, il est possible, lors du déploiement de l'application, de compiler une release pour *Windows*, et une pour *Linux*. Pour *Windows*, seule une poignée (17) de fichiers *DLL* (des bibliothèques windows) sont requis pour faire fonctionner l'application, et ils sont fournis dans l'archive du lien de téléchargement. En revanche pour *Linux*, il est nécessaire d'avoir installé a minima le paquet *qt5-default* (pour *Ubuntu/Debian*).

En addition de ce framework, nous avons utilisé l'IDE (*Integrated Development Environment* dédié à *Qt* : *Qt Creator*). Cet IDE s'utilise un peu comme Microsoft Visual Studio C++, c'est-à-dire que son gros atout réside en la possibilité de concevoir les parties graphiques (appelés *Widgets*, cela comprend les boutons, les labels, les line-edit, les barres de progression, etc...) directement depuis l'IDE en plaçant à la souris les éléments sur les différentes fenêtres de l'application. Cela est rendu possible par son outil intégré appelé *Qt Designer* (c.f. **Image 8**).

Image 8: Qt Designer, l'outil de Qt Creator pour créer simplement des interfaces graphiques



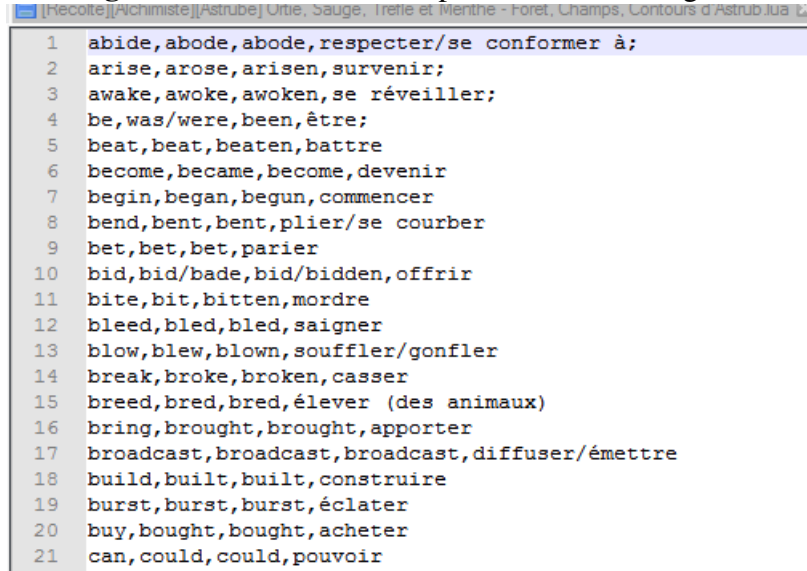
Bien que la programmation brute ait été largement allégée par ce procédé, il a quand même fallu implémenter le moteur de l'application. Comme vous pouvez le voir sur l'**Image 8**, tous les *Widgets* se superposent indifféremment sur la fenêtre. Il faut dire à l'application quand ils doivent apparaître, disparaître, afficher telle ou telle valeur ou chaîne de caractères.

Pour réaliser le moteur de production de verbes irréguliers et mots de vocabulaire, nous avons dû concevoir un générateur aléatoire à exclusion; c'est-à-dire un générateur aléatoire qui exclut de sa génération les mots ou verbes déjà tirés. Pour cela nous avons créé une petite fonction *randAB(int min, int max)* qui génère aléatoirement un entier entre deux bornes passées en paramètres. Cette fonction est ensuite utilisée pour générer aléatoirement un indice du tableau de mots/verbes chargé en mémoire. Suite à quoi cet indice est utilisé pour piocher dans ce tableau de mots/verbs, puis il est ajouté à un tableau d'exclusion (de type tableau d'entier). Chaque fois qu'un nouvel indice est généré, on vérifie qu'il n'existe pas déjà dans ce tableau d'exclusion. Si il a déjà été tiré, un nouvel indice est généré, sinon il est utilisé pour piocher dans le tableau de mots/verbes, et ainsi de suite. Lorsque la taille du tableau d'exclusion atteint le nombre de ligne du tableau de mots/verbes, on en déduit que c'est la fin de l'exercice.

Afin de réaliser un travail d'équipe efficace, nous avons utilisé le gestionnaire de version **git**, extrêmement répandu et maîtrisé par nous-même. Nous avons hébergé notre dépôt sur [GitHub](#). Le projet est d'ailleurs placé sous une Licence Public Générale GNU v3.0.

En ce qui concerne les différentes bases de données, nous avons tenu compte du fait que ces dernières étaient plutôt simples et légères. De plus, nous voulions appuyer sur la partie "personnalisable" des séances de révision. Il nous fallait donc une structure de donnée simple. Elle fut toute trouvée : le format CSV. Très répandue, elle a également l'avantage d'être facile de conception ou de modification. LibreOffice ou Microsoft Excel voire un simple éditeur de texte suffisent pour cela.

Image 9: Structure d'un CSV pour les verbes irréguliers



1	abide,abode,abode,respecter/se conformer à;
2	arise,arose,arisen,survenir;
3	awake,awoke,awoken,se réveiller;
4	be,was/were,been,être;
5	beat,beat,beaten,battre
6	become,became,become,devenir
7	begin,began,begun,commencer
8	bend,bent,bent,plier/se courber
9	bet,bet,bet,parier
10	bid,bid/bade,bid/bidden,offrir
11	bite,bite,bitten,mordre
12	bleed,bled,bled,saigner
13	blow,blew,blown,souffler/gonfler
14	break,broke,broken,casser
15	breed,bred,bred,élever (des animaux)
16	bring,brought,brought,apporter
17	broadcast,broadcast,broadcast,diffuser/émettre
18	build,built,built,construire
19	burst,burst,burst,éclater
20	buy,bought,bought,acheter
21	can,could,could,pouvoir

Pour construire une base de données adaptée aux verbes irréguliers, il suffit de mettre sur chaque ligne les trois formes et la traduction, sans espaces, et séparés par des virgules (c.f. **Image 9**). La forme de chaque ligne est donc la suivante :

infinitif,preterit,participePasse,traduction

La base de données des mots de vocabulaire est plus simple. Simplement le mot anglais et la traduction française séparés par une virgule sur chaque ligne. La forme d'une ligne est donc la suivante :

motAnglais,traductionFr

Difficultés rencontrées

La première difficulté à laquelle nous avons été rapidement confrontés a été la nouveauté du langage et du framework. Le C++ n'est pas réputé pour être un langage facile d'appréhension. En plus de cela il a fallu ingérer une quantité relativement important de documentation technique sur le framework *Qt* et sur l'IDE *Qt Creator*. Il fut plus facile de gérer l'IDE, qui est conçu pour être ergonomique.

Le C++ a dû être utilisé dans sa forme Orientée Objet. Bien que nous soyons bien formés pour ce type de paradigme de programmation, notamment grâce au *Java*, les différences de langage, de sémantiques, et de conceptions nous ont posé problème tout le long du développement. Nous avons néanmoins réussie à surmonter ces épreuves, notamment grâce à nos connaissances en C.

Devant la colossale charge de travail nous avons décidé de séparer intelligemment le code en plusieurs parties. Une première partie gère la fenêtre d'accueil. Deux autres parties gèrent indépendamment la fenêtre de vocabulaire et la fenêtre de verbes irréguliers. Enfin, une dernière partie, qui elle n'est pas orienté objet (n'est pas une classe, se comporte comme un fichier C, langage que nous maîtrisons), qui fournit plusieurs fonctions utiles aux trois autres parties. Ce ne fut pas une tâche facile pour des novices en la matière tels que nous.

Une autre difficulté a été de réussir à créer le générateur aléatoire à exclusion. Beaucoup d'erreurs à l'exécution (des *overflow*), dues à des mauvaises et hasardeuses indexations, nous ont frenés dans le développement.

Il a également été difficile de construire une base de données pour le vocabulaire qui ne soit pas [trop] ambiguë. La polysémie des mots augment de façon exponentielle la difficulté du problème. De ce fait cette base de données ne fait actuellement que 40 lignes (soit 40 traduction possibles). Nous avons été obligé de conserver un moteur de vérification simple pour ne pas faire exploser la densité du code, et la complexité en mémoire et en temps à l'exécution.

Pour finir, nous avons dû créer un parseur de fichiers CSV (nos bases de données). Ne connaissant pas bien le C++ avec le framework *Qt*, il fut plutôt difficile d'implémenter correctement ce parseur. Ce parseur s'étend également à toutes les line-edit; cela consiste en l'élimination des virgules et des espaces, pour ne pas corrompre le fichier des sauvegarde lors du *Save and Quit*. Rappelons qu'une virgule marque la séparation des éléments dans un CSV.

Conclusion

The fact that the topic allowed us a free design for the application was a real boon for us. Therefore we could break the new, using a novel framework, and a legendary language. But it had a cost, it took us a long time to lead to something almost functional.

Thus, it is undeniable that this new knowledge is a strong advantage for us and, maybe, our future career. Moreover, it allowed to check our basics about English language, and this was significant for us. But the main point here is that we build a powerful tool, made for students, really portable, easy to use. It could help beginners, or confirmed users who need to check their bases as well. So for us, being able to custom your own sessions is the most substantial point of our app.

For now, our app is in version 1.1, it is stable, but quite simple. in the future we have considered to improve our app by four ways. First, to improve the provided databases, above all the vocabulary: 200 seems to be a quite good amount. Secondly, to alternate French and English translation in the vocabulary exercise. Thirdly, to unlock translation of irregular verbs in the verbs exercise (it is blocked for the moment, due to the last way). And finally, to improve our validation engine in order to allow polysemous words, but this is a big challenge, due to the mathematical problem exposed in the previous chapter.

Appendix

Bibliography

<http://doc.qt.io/qt-5/>

<https://openclassrooms.com/>

<https://stackoverflow.com/>

Our initial database of irregular verbs: <https://verbes-irreguliers-anglais.fr/>

<https://translate.google.fr/?hl=fr>

<https://en.wikibooks.org/wiki/LaTeX>

<https://tex.stackexchange.com/>

https://en.wikipedia.org/wiki/English_irregular_verbs

Download links

Archive with executable (Win and Linux):

<https://github.com/AOSaaron/learnEnglish-app/releases/tag/learnEnglish-v1.1>

Sources:

<https://github.com/AOSaaron/learnEnglish-app>