Julien VAUBOURG
Romain CARETTE

ESIAL – 2A 2012

UNIVERSITÉ DE LORRAINE

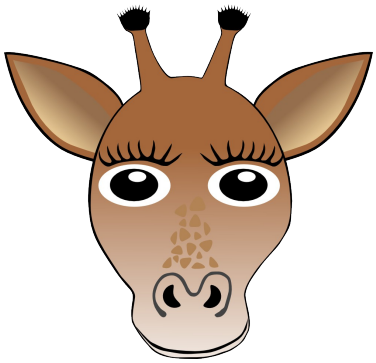
École associée de INSTITUT TELECOM

esial

ÉCOLE SUPÉRIEURE D'INFORMATIQUE ET APPLICATIONS DE LORRAINE

Summary

User documentation



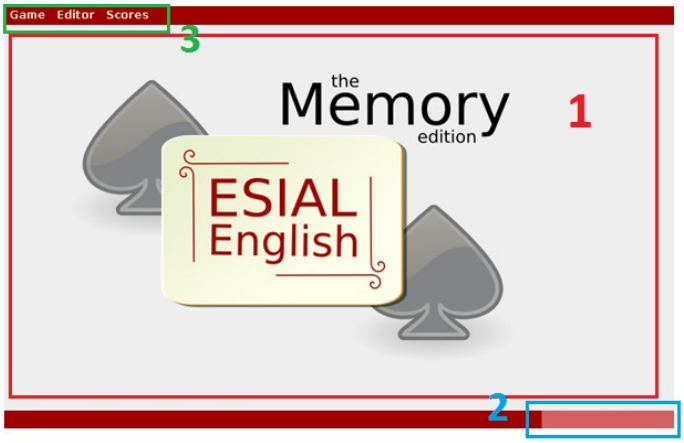
Documentation technique

User documentation

1. Preamble

We decided to build a game based on English vocabulary so that people can learn English while entertaining themselves. First we will present the main interface and its different game options. After that, we will explain you how to start a game. Then there will be the features, the game policy and the scores. And finally we will see the editor guide.

2. Main Interface



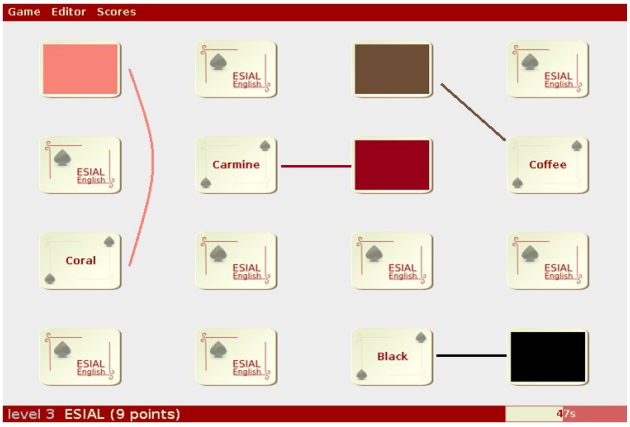
In the interface we have three parts :

- in the middle, we have the playground. *(1)*
- at the bottom right, we have a timer. *(2)*
- at the top, we have a menu bar which contains game, editor and scores menus. *(3)*

3. Game options

The aim is always to associate a card to another. We have implemented five different game modes which are the following : *Colors*, *Numbers*, *Pictures*, *Synonyms* and *Translations*.

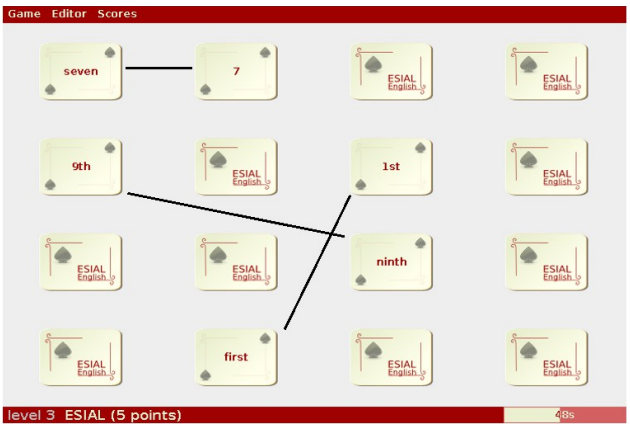
3.1 Colors



The first one is colors as people usually learn them when they are young and it's normally easier. Players have to associate a color to its name. We have linked the element in the example.

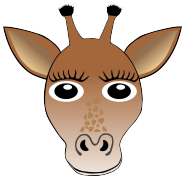
3.2 Numbers

42



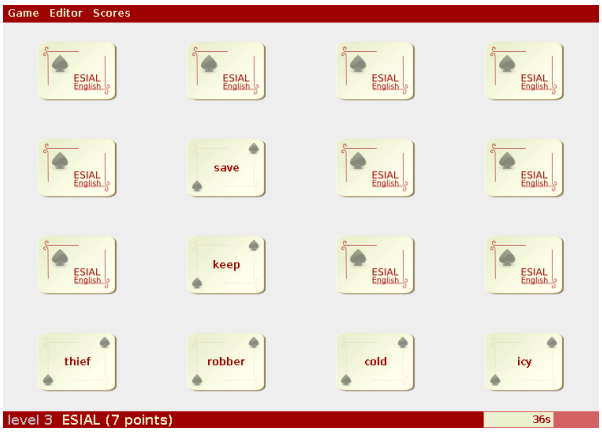
The second mode is based on numbers for the same reason than the first one. Users have to find a number or an ordinal number and its denomination.

3.3 Pictures



The third mode deals with pictures and what is represented. It can help people to memories the names by associating them to their representations.

3.4 Synonyms



We also have a mode which can be used to extend the known vocabulary because with this option synonyms have to be found.

3.5 Translations

In the last one users can improve their translation skill with the so-named mode.

4. How to start a game

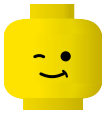
In the game menu we have the five choices we have just seen.

So in order to start a game users have to pick one and that will bring them to the starting panel which is the following :

In this frame players can choose their starting level and their names. When this is done, the chosen mode begins.

5. Game guide

5.1 Features

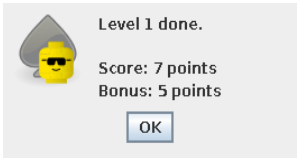


During the game, users can view various messages, corresponding to game features.

Each game is timed, so the player can be timed out if he is too slow to end the game.

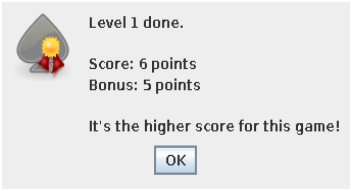
When his or hers time is up, the game is over.

But, if the user finds all pairs within the time, he will access the next level. Here we can see that a new level gives him five points.



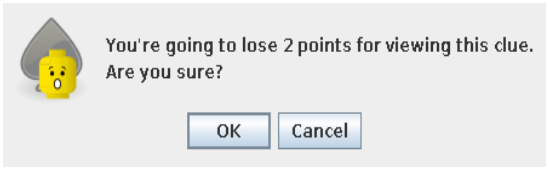
If the user both plays in time and has a higher score than the registered one he is notified.

Because this is an educational game, the player can have a clue if he clicks twice on the card. Clues are definitions or translations according to the current game.



This part permits the user to learn new words, and as a result, he can have a better score next time.

The more he plays, the more vocabulary and good score he gets.



Here is a example of clue one could get when playing with the translations game. It gives the definition of a word.

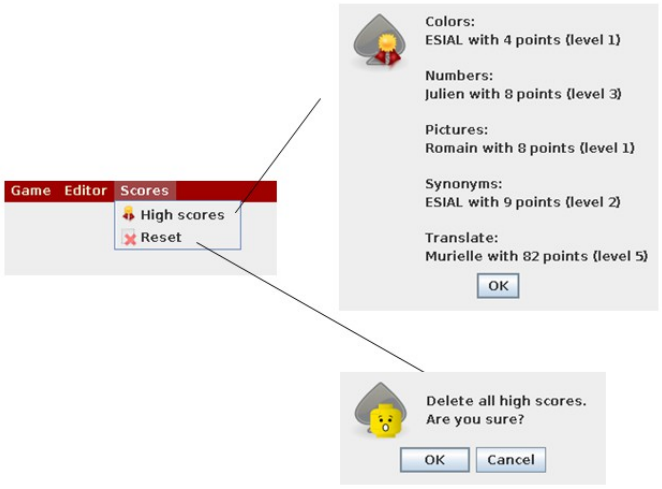
5.2 Game policy

When the player discloses a good pair of cards, he gains one point. But when he discovers a wrong pair, he loses the same amount.

Consulting a clue costs two points. And as a user can lost many points with a single level, ten bonus points for each new level are granted. Moreover he starts with ten points. Points can be negative.

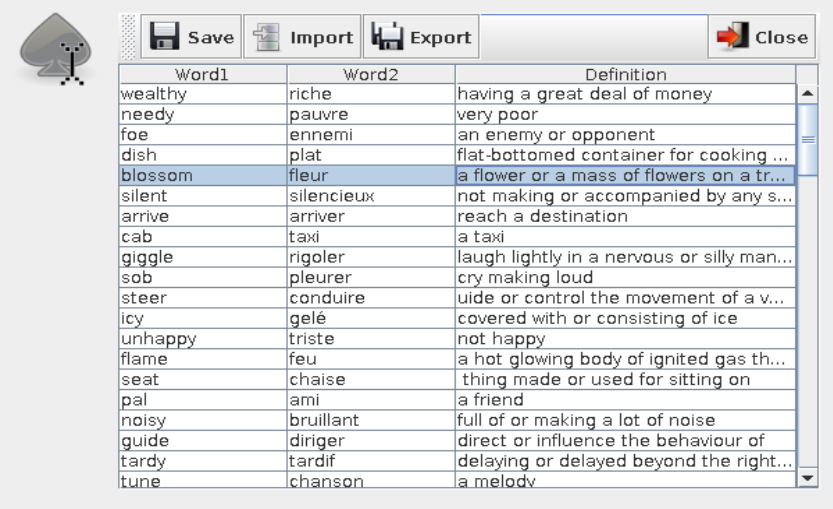
For each new level, the number of cards is 2 power the level, and the timer is : thirty seconds multiplied by the level.

5.3 Scores



We found of paramount importance to have a sort of race. So, the user will be challenged and will want to improve himself. For this purpose, the top score for each game is registered in the high scores window. He can reset it when he wants.

6. Editor guide

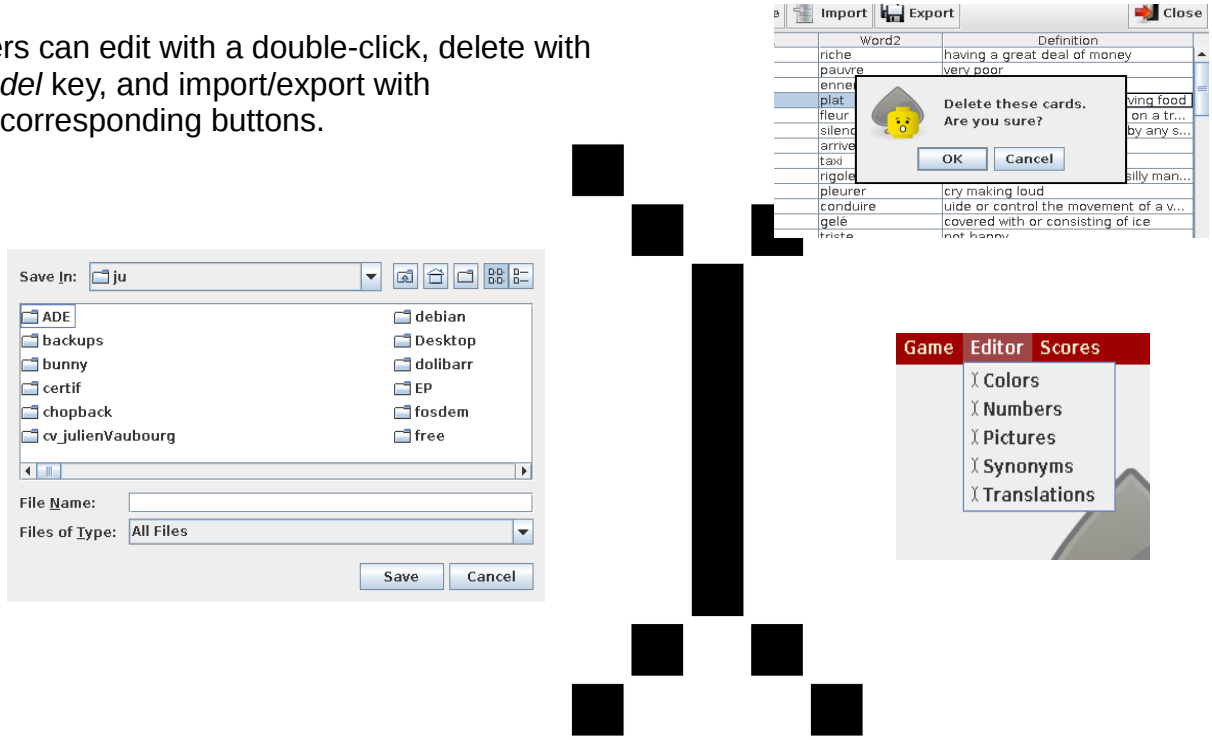


Word1	Word2	Definition
wealthy	riche	having a great deal of money
needy	pauvre	very poor
foe	ennemi	an enemy or opponent
dish	plat	flat-bottomed container for cooking ...
blossom	fleur	a flower or a mass of flowers on a tr...
silent	silencieux	not making or accompanied by any s...
arrive	arriver	reach a destination
cab	taxi	a taxi
giggle	rigoler	laugh lightly in a nervous or silly man...
sob	pleurer	cry making loud
steer	conduire	uide or control the movement of a v...
icy	gelé	covered with or consisting of ice
unhappy	triste	not happy
flame	feu	a hot glowing body of ignited gas th...
seat	chaise	thing made or used for sitting on
pal	ami	a friend
noisy	bruillant	full of or making a lot of noise
guide	diriger	direct or influence the behaviour of
tardy	tardif	delaying or delayed beyond the right...
tune	chanson	a melody

Because a static game is not useful for the future, our memory is scalable. Everyone, without computer skills, can add, edit or remove words of the databases. People as to choose the database they want to edit in the editor menu.

In order to encourage collaboration, a user can import or export his modifications, into a simple text file which is easy to exchange.

Users can edit with a double-click, delete with the *del* key, and import/export with the corresponding buttons.



7. Why this game rocks

Our Memory is user-friendly, with its nice graphic interface and is easy to use. It has different modes in order to avoid the player to be bored.

It is entertaining with the race possibilities, but educational with the clues which will permit him to learn new words and the aim of the game.

And finally it is scalable, with a nice and complete editor that add a community goal.

Documentation technique

1. Patrons de conception

Dans le cadre de ce projet, nous avons essayé de mettre en oeuvre le maximum de bonnes pratiques apprises durant nos (presque) deux ans d'école.

Ainsi, nous avons utilisé un patron de conception (*design pattern*) qui correspond au *modèle-vue-contrôleur* (MVC). Techniquement, ce sont les classes Java qui correspondent aux modèles qui contiennent le coeur du jeu : les fonctions de jeu (créer le plateau, trier les cartes, indiquer si la paire retournée est valide, etc.), ainsi que les données (base de données des cartes, jeu en cours, scores, etc.).

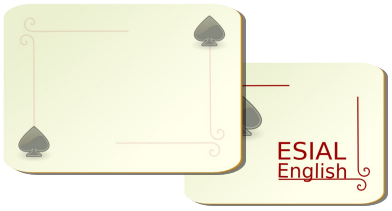
Les vues s'occupent de tout ce qui affichage, ce qu'on appelle l'interface homme-machine. Ce sont elles qui construisent l'interface graphique du jeu, proposent les menus, affichent les cartes en fonction des données fournies par les modèles, etc. Enfin, les classes qui correspondent aux contrôleurs ont pour fonction de s'occuper de l'interaction entre les deux. Lorsque le joueur clique sur une carte, il déclenche le contrôleur qui y est associé, pour modifier le modèle. Celui-ci met alors à jour ses données, et prévient la vue qu'il y a eu du changement. La vue met l'affichage à jour, la carte se retourne, la boucle est bouclée.

La mise à jour des vues se fait avec le patron observeurs, qui impose aux vues de se mettre à jour sur modèle, plutôt que ce soit celui-ci qui soit contraint de les avertir. Lorsque le modèle enregistre une modification dans ses données, il averti ses observeurs qu'il y a eu du changement. Selon le paramètre qu'il passe au moment de cette alerte, toutes les vues s'activent pour rafraichir leur affichage.

Enfin, nous avons utilisé le patron Commandes pour la gestion du menu, ce qui nous a imposé une programmation stricte et ordonnée, pour arriver à un résultat stable.

Cette façon de faire nous a permis d'écrire efficacement un programme, de façon propre, pas à pas. Il pourra ainsi facilement évoluer, et acceptera des ajouts de fonctionnalités sans mettre la base en péril, tant que la rigueur de cette façon de faire sera au rendez-vous.

2. Ergonomie



D'un point de vue ergonomie, le jeu devait être réactif : les cartes devaient se retourner suffisamment rapidement, mais rester affichées suffisamment longtemps pour que le joueur

puisse identifier la nouvelle carte. Pour arriver à quelque chose de réactif, nous avons utilisé des *threads*. Le chronomètre, lui aussi, utilise un *thread* indépendant.

Afin de réussir à afficher des images sur des cartes, en gardant le jeu évolutif, nous avons utilisé le caractère booléen des options d'activation et de désactivation des boutons. Ainsi, une carte qui se retourne correspond à un bouton de Swing qui se désactive. En jouant sur les propriétés graphiques de l'état désactivé des boutons, nous avons pu lui associer l'image des cartes retournées. Le bouton étant désactivé, l'avantage de cette stratégie est que l'utilisateur ne peut plus appeler le contrôleur des cartes en cliquant de nouveau dessus. S'il s'agit de la première carte, un nouveau contrôleur a été attribué dans ce contexte : celui de l'affichage des indices.

L'affichage des mots sur les boutons se fait habituellement avec la valeur du bouton. Pour éviter d'avoir à remplacer totalement l'image de la carte retournée par une image du jeu lorqu'on est en mode Images, nous avons surchargé le bouton avec un label. Le label ayant lui-même des fonctionnalités d'activation et de désactivation, nous les avons exploitées pour afficher l'image à retrouver à l'état désactivé. En ayant pris soin de détourer chacun des images du jeu, elles apparaissent alors sur le fond de carte retournée.

3. Un éditeur puissant et intuitif

L'éditeur a principalement été réalisé à l'aide du composant *JTable* de Swing. Il s'agit d'un atout intéressant de Java pour traiter des données qui peuvent se présenter sous forme de tableau. Cependant, les fonctionnalités sont limitées : pas de fonctionnalités d'ajouts interactifs, ni de suppression, et l'édition demande à bien le configurer. Nous avons alors décidé de réécrire complétement le modèle du composant, pour qu'il s'affiche en allant chercher lui-même les données dans les fichiers textes. Nous avons pris soin de concevoir la structure de ceux-ci en respectant le standard CSV (*Coma Separated Values*). Cette solution nous permet d'obtenir des fichiers faciles à diffuser, et à modifier, y compris sans éditeur, par un utilisateur lambda. Dans le cadre de l'éditeur, il devient facile à parser et à exploiter.

Pour l'édition intéractive des champs, nous avons pris soin d'associer un contrôleur à chacune des cases du tableau, pour lui apprendre à enregistrer automatiquement, et en pleine autonomie, les modifications directement le vecteur des données. Au moment où l'utilisateur clique sur le bouton de sauvegarde, le contenu du fichier est synchronisé avec le vecteur.

Dans un même ordre d'idée, la suppression n'est pas du tout gérée nativement. Ainsi, nous avons été contraints d'associer la touche suppression des claviers à un événement qui déclenche la confirmation de suppression. Puis, la ligne est supprimée du composant graphique, ce qui a pour conséquence de modifier le vecteur de données au même instant.

Afin d'améliorer la navigation dans l'éditeur, nous avons modifié le comportement de la navigation dans le tableau, en s'assurant que le curseur passe soit au champ suivant, soit au premier champ de la ligne suivante, lorsqu'un champ vient d'être édité. Si c'est le dernier

champ de la dernière ligne qui a été édité (cas d'un ajout de définition), une nouvelle ligne apparait pour permettre un ajout théoriquement infini de définitions à la suite.

4. Les bases de données

Les bases de données sont introuvables en l'état sur la Toile. Nous avons été contraints de rassembler nous-mêmes les définitions, en utilisant le site *WordReference.com*, *synonym.com* ainsi qu'un traducteur disponible sous Android.

La banque d'images a été trouvée sur Google Images, en tapant des mots au hasard, jusqu'à obtenir suffisament d'images pour constituer une base de données exploitable. Chacune des images a été retraitée par nos soins, pour lui permettre d'être au format des cartes, et leur retirer les fonds de couleur.

Nous avons rencontré quelques problèmes d'encodage vis-à-vis de l'encodage par défaut de Windows et celui de GNU/Linux. Parce que nous ne travaillons pas tous les deux sur le même système d'exploitation, le problème s'est posé plusieurs fois.

5. Esthétisme du produit

Puisqu'il s'agit d'un produit qui doit être accessible à un public qui n'est pas informaticien, nous avons essayé de profiter de ce projet pour aller au bout du développement, en s'attardant sur l'aspect esthétique. C'est en effet quelque chose qu'on ne nous demande jamais, et qui n'est jamais valorisé.

Ainsi, nous avons passé du temps du trouver un jeu de couleurs et une cohérence graphique. Par exemple, chacune des icônes des messages ou des fenêtres a été remplacée par une icône spécifique au logiciel. Nous avons créé toutes ces icônes, à partir d'une bibliothèque d'images vectorielles libres, à l'aide du logiciel libre Inkscape. Les cartes ainsi que le reste des images ont aussi été créés par nos soins, pour ce projet.

Dans le même ordre d'idée, nous avons essayé de styliser des modules graphiques de l'interface. Nous avons utilisé les possibilités de modifications qu'offre Swing (le gestionnaire graphique que nous avons couplé avec Java), à cet effet. Malheureusement, il s'agit de quelque chose de très peu documenté, et qui a demandé beaucoup de recherches, pour arriver à un résultat satisfaisant. À titre d'exemple, nous avions pour ambition de redessiner tout le menu principal du jeu, qui est un composant Swing à part entière. Après avoir passé plusieurs heures à trouver les noms des propriétés qui correspondaient aux différentes parties graphique, et à choisir des couleurs cohérentes, nous n'avons pas trouvé de solution pour redessiner les contours bleus des sous-menus. Cette mésaventure nous a valu de revenir un pas en arrière pour garder un ensemble esthétique.

Ce projet nous a permis d'explorer d'autres façons de concevoir un logiciel, tout en nous interrogeant sur les possibilités pédagogiques qu'un tel jeu peut offrir.