



**Your new way to learn English:
Eas'Improve**

2A - promo 2018
Projet informatique individuel

Rapport du projet informatique individuel

27 avril 2017

Noémie NEYRON

Sommaire

INTRODUCTION : RAPPEL DU PROJET.....	2
Rappel de l'objectif	
Ressources disponibles	
Choix de conception, demande client	
Choix de technologie	
 ÉTAPES DU PROJET, GESTION DE PROJET.....	 5
Apprentissage JavaScript	
Partie 1	
Partie 2	
Gestion des bases de données	
 PRESENTATION DES RESULTATS.....	 7
Architecture logicielle	
Sessions de jeu	
Fonctionnalités	
Bases de données	
Explication technique	
Problèmes rencontrés	
 TESTS	 20
 INTEGRATION DES JEUX DEVELOPPES DANS LE SITE EAS'IMPROVE.....	 21

Introduction : rappel du projet

Objectif

L'objectif du projet était de développer deux jeux en ligne en collaboration avec Mme Tracy Carmona, professeur d'anglais, et le groupe de projet Transdisciplinaire Eas'Improve. Ces jeux doivent aider les étudiants de l'ENSC à améliorer leur anglais de manière la plus ludique et efficace possible, et doivent être accessible sur le site créé dans le cadre du projet par les élèves de première année.

Ressources disponibles

Les élèves du groupe de transdisciplinaire avec lesquelles j'ai travaillé m'ont donné accès à leurs documents et aux résultats de quatre mois de travail de réflexion sur les *serious games* dédiés à l'apprentissage des langues.

Ainsi, j'ai eu accès à un Benchmark des différents sites et applications qui visent à aider leurs utilisateurs à améliorer leurs compétences en langues étrangères, par exemple *Englishclub* ou *Usingenglish* pour les sites et *Babbel* ou *Duolingo* pour les applications.

De plus, leur état de l'art regroupe des recherches sur le fonctionnement de la mémoire, sur les différents types de mémoire ainsi que les techniques de mémorisation.

Le concept des jeux que j'ai réalisés était leur idée, à laquelle j'ai ajouté quelques suggestions et adaptations.

Choix de conception, demande client, communication client

Le concept des jeux a été discuté avec le groupe de transdisciplinaire et les choix ont été faits avec leur accord. Nous avons ainsi réfléchi à des jeux aux performances optimales en s'appuyant sur les études sur la mémoire précédemment citées. Du Benchmark est ressorti que leur site, et donc, par extension, les jeux que je devais coder, devaient être les plus divertissants et esthétiques possibles, gratuits et surtout, opérer un renforcement de la mémorisation en faisant appel successivement à différents types de rappel. Ainsi, nous avons choisi de partir sur deux jeux, chacun basé sur un de ces axes :

- 🌂 Le premier jeu est un jeu de vocabulaire, qui s'appuie sur une base de données de verbes à particules, qui posent souvent problème à ceux qui apprennent l'anglais. Il est basé sur le principe des flashcards, selon la demande de Mme

Carmona. Le jeu comporte trois étapes, qui doivent renforcer la mémorisation.

- 🧑 La première étape est un affichage simple des flashcards : de chaque côté de la carte se trouve une expression et sa traduction. L'utilisateur peut retourner la carte autant de fois qu'il le veut, de façon à apprendre l'expression.
- 🧑 La deuxième étape fait appel à la mémoire indicée. Cela signifie qu'on ne lui demande pas directement de traduire une expression, mais de relier un mot à sa traduction dans un jeu inspiré du Memory : les cartes sont rangées dans un tableau et le joueur doit les appairer.
- 🧑 La troisième étape permet de renforcer la connaissance tout en s'assurant que l'utilisateur est capable de traduire une expression, et pas seulement de reconnaître les paires. On lui demande ainsi une traduction directe de l'expression qui s'affiche dans la carte.

Les points ne sont comptés que pour les deux dernières étapes : pour les appariements, l'utilisateur démarre avec un score de 10/10 et perd un point à chaque erreur. S'il fait trop d'erreurs et arrive à zéro, il doit retourner à l'étape des flashcards, afin de mieux apprendre les expressions. Lors de la troisième étape, chaque réponse juste apporte un point. À la fin des trois étapes, le joueur peut rejouer, avec de nouvelles expressions choisies au hasard.

- 🧑 Le second jeu est principalement axé sur le côté ludique de l'apprentissage, qui permet également à l'utilisateur de mieux retenir les informations. Il s'agit d'un jeu graphique dans lequel un utilisateur peut déplacer une fusée à l'aide de sa souris, et tirer d'un clic. Des vaisseaux étrangers, portant différentes expressions en anglais, sont présents au-dessus de la fusée, et l'utilisateur ne doit tirer que sur les expressions qui sont incorrectes grammaticalement. Ce jeu permet aux utilisateurs de reconnaître comme telles des erreurs qui sont faites régulièrement.

Dans ce second jeu, les points sont comptés en fonction du nombre d'erreurs et du temps. Pour chaque bonne réponse, c'est-à-dire pour chaque expression fautive explosée, l'utilisateur marque six points. À chaque fois qu'il tire sur une expression qui est correcte grammaticalement, il perd deux points. Enfin, il perd un point à chaque seconde.



Le groupe de transdisciplinaire avait également établi une charte graphique en sélectionnant six couleurs principales. Je m'en suis servie pour le premier jeu, mais pas pour le second qui se base uniquement sur des images d'espace.

Les résultats de mon travail ont été transmis à Mme Carmona et au groupe de transdisciplinaire notamment par l'intermédiaire d'un *pen* sur le site CodePen. Les contacts avec le groupe de transdisciplinaire ont été réguliers, que ce soit par mail ou de visu.

Choix de technologie

Choix du JavaScript

Le projet devant être réalisé pour le web, le langage qui a paru le plus approprié est JavaScript. En effet, il m'a paru être un langage Web puissant et utile pour mon avenir, et m'a semblé bien convenir aux différents enjeux de mon projet, que ce soit la partie plus algorithmique du premier jeu ou le côté graphique du second.

Ayant préféré me concentrer sur un apprentissage approfondi du JavaScript et sur le bon fonctionnement des jeux, plutôt que de me disperser dans de multiples langages et techniques, j'ai d'abord décidé de ne pas utiliser jQuery.

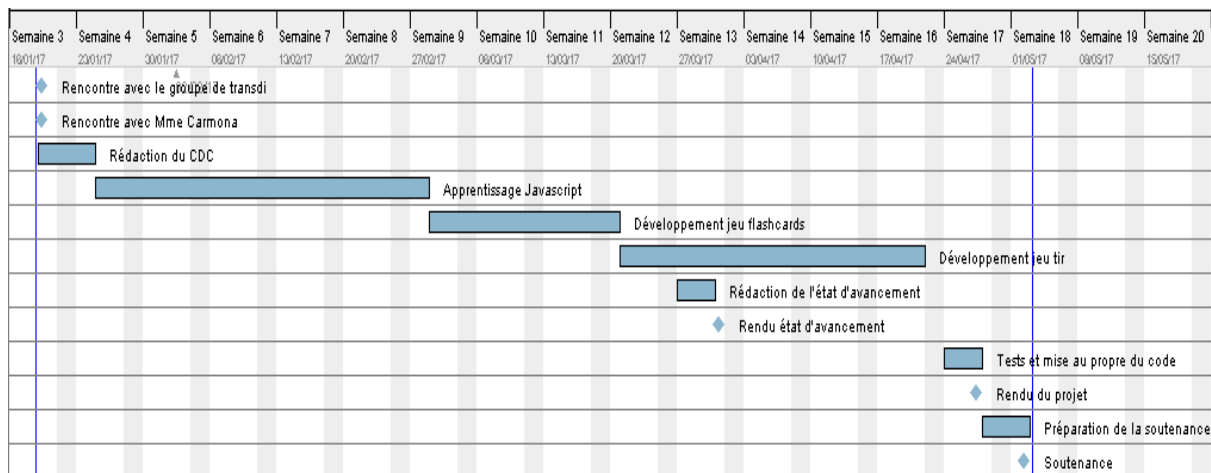
Cependant, vers la fin du projet, je me sentais plus à l'aise avec le JavaScript, et j'ai eu envie de faire disparaître une image progressivement. J'aurais pu réaliser cela en JavaScript seul, par exemple avec un timer qui générerait cette disparition, mais la fonction *FadeOut* de jQuery, bien plus simple d'utilisation, m'a convaincue de m'essayer à cette bibliothèque. J'ai bien conscience qu'utiliser jQuery pour cela seulement est sous-optimal, puisque cela ajoute de la consommation et des données à charger, mais il s'agissait simplement d'essayer cette technologie, dont j'aurais pu tout à fait me passer. Je pourrais donc fournir au groupe de transdisciplinaire une version sans cet ajout, s'il me le demandait.

Le logiciel utilisé pour coder en JavaScript est Visual Studio Code, qui nous a paru approprié pour les différents langages Web utilisés (HTML, CSS, JavaScript).

Gestion des données

Nous avons choisi d'effectuer la gestion des données à l'aide d'une API, afin de faire le lien entre les données du groupe de transdisciplinaire et mes jeux qui peuvent être indépendants du site Eas'Improve. Cet API permet donc de créer un pont entre la base de données, accessible en PHP-SQL à l'aide de requêtes GET notamment, et mon programme codé en JavaScript. Ce pont s'effectue par l'intermédiaire du langage JSON, qui permet d'encoder les données SQL et qui est lisible par JavaScript grâce à une requête AJAX et la fonction de décodage `JSON.parse()`.

Étapes du projet, gestion de projet



Planning initial

Apprentissage du langage

Au cours de la première partie du projet, je me suis exercée à utiliser le langage JavaScript, que je ne connaissais pas. Pour cela, j'ai effectué des tutoriels sur OpenClassrooms sur le fonctionnement de ce langage, dans un versant computationnel mais aussi graphique avec l'élément canvas notamment.

Liste non exhaustive des cours OpenClassrooms utilisés :

- 👉 Créez des pages Web interactives avec JavaScript
- 👉 Tout sur le JavaScript !
- 👉 Créer un mini-RPG en JavaScript avec Canvas

J'ai dû également relire certains tutoriels sur le fonctionnement du HTML, dont je me sers aussi comme base à laquelle appliquer mon code en JavaScript. De plus, les forums et les documentations m'ont été bien utiles, que ce soit en JavaScript, en HTML ou en CSS.

Sur les conseils de mon tuteur, j'ai continué à me former au langage en commençant directement le premier jeu à créer, afin de créer un code fonctionnel, quoique pas toujours optimisé en terme de redondance notamment. Ce code a été légèrement repris depuis mais resterait probablement à optimiser, quoiqu'il soit tout à fait fonctionnel.

Code de la première partie

Conformément à mon planning, le premier jeu, constitué de trois étapes, a été réalisé en premier, et a été l'occasion de m'exercer au JavaScript. Le code de cette partie, qui devait m'occuper lors des trois premières semaines de mars, a été commencé

avant, et fini dans les temps.

Cependant, quelques fonctionnalités ont été rajoutées à la fin du projet, notamment la possibilité de perdre à la fin de la deuxième étape et de devoir recommencer, ou celle de rejouer à la fin des trois étapes.

Code de la seconde partie

La seconde partie m'a occupée de la mi-mars à la troisième semaine d'avril. Le déplacement de la fusée a été codé en premier, avec l'affichage de quatre propositions seulement. Après réflexion, j'ai décidé de modifier ce fonctionnement : au lieu d'afficher quatre propositions au hasard parmi les dix, dans un tableau en haut de l'écran, j'ai décidé d'utiliser dix canvas différents, ce qui m'a permis d'une part de placer chaque vaisseau étranger à une hauteur différente, et d'autre part de bien mieux gérer l'affichage du faisceau laser du tir jusqu'au vaisseau visé.

Gestion des données

La gestion des données est venue plutôt tard dans le projet : elle a été gérée vers la mi-avril, alors que les principales fonctionnalités des différents jeux étaient déjà en place à partir de tableaux entrés « en dur » dans le code.

Cela s'explique notamment par la multiplicité des acteurs liés à ces bases de données : T. Carmona, qui a rédigé les entrées, le groupe de transdisciplinaire qui en a fait des bases de données formelles, B. Pesquet qui a créé et hébergé l'API qui relie la base de données en SQL du groupe de transdisciplinaire au fichier JavaScript de mon projet, par l'intermédiaire du langage JSON.

Gestion de projet

Le Gantt initial a été globalement suivi, avec les aménagements qu'on a évoqués sur l'apprentissage et le code simultanés. De plus la gestion de projet a aussi été réalisée à l'aide d'un tableau Trello, partagé avec mon tuteur, qui m'a permis de lister les tâches à effectuer.

Présentation des résultats

Architecture logicielle

Mon projet ne contenant pas de fichiers multiples pour chaque jeu, ni de classes, on ne peut pas à évoquer une architecture de pages ou de classes. Les deux jeux étant totalement indépendants, j'évoquerai leur hiérarchie séparément.

Premier jeu : Flashcards

Ce jeu s'organise en trois étapes, autour de deux éléments HTML fondamentaux : un bouton *boutonCarte*, qui est la flashcard, et un tableau *tableau* qui contient les cartes pour l'étape 2. Les données extraites de l'API sont stockées dans un tableau *tableauVerbes* qui contient deux fois plus de cases que d'entrées dans la base : dans toutes les cases paires, j'ai rangé les propositions en anglais, et dans les cases impaires les équivalents en français. Chaque expression anglaise est donc associée à l'expression française qui la suit directement.

Ce tableau est secondé par le tableau *tableauCartes* qui contient les dix expressions qui seront effectivement utilisées au cours de la partie. Pour les mêmes raisons que précédemment, ce tableau possède vingt cases. Un troisième tableau *tableauQuestionsAPoser*, permet de stocker le numéro des questions à poser. Ce tableau n'est pas vraiment indispensable, mais il simplifie la gestion des doublons.

Pour fonctionner, le jeu a également besoin de quelques variables globales qui permettent de sauvegarder les scores (*scoreEtape2*, *scoreEtape3*), le jeu en cours, le numéro de question, quelques booléen qui me permettent de spécifier les réactions du programme, comme *rejouer* qui permet de savoir si on rejoue la première étape avec le même *tableauCartes* ou non. L'étape 2 a besoin de nouvelles variables globales, qui doivent donc « survivre » à plusieurs appels de la fonction *verifierCaseCliquer()* : les deux cases cliquées au tour précédent, qui renvoient directement aux objets HTML que sont les cases du tableau HTML, le booléen *reponsePrecedenteJuste* qui permet également de tenir compte du tour précédent, et les variables *nombreCartesColorees* et *nombreQuestionsRepondues* qui permettent respectivement de savoir si la carte cliquée est la première ou la seconde à l'être, et si l'utilisateur a éliminé toutes les paires.

Les autres variables globales sont les variables qui correspondent aux objets HTML que j'ai besoin de manipuler dans mon programme.

De nombreux objets HTML portent des événements *onclick* JavaScript :

- 👉 les cases du tableau,
- 👉 le bouton *boutonCarte*, qui déclenche la fonction *retournerCarte()*, qui sert dans l'étape 1 et l'étape 3,

- 🌈 les boutons *jeNeSaisPas* et *boutonValider*, qui déclenchent la fonction *verifierReponse()*. Si l'utilisateur a indiqué qu'il ne savait pas, on vérifie quand même ce qu'il a écrit. Généralement il n'a rien écrit, mais s'il a indiqué la bonne réponse, elle lui est comptée comme telle.
- 🌈 *boutonQuestionSuivante*, qui n'est visible qu'aux étapes 1 et 3 et déclenche la fonction *tirerNouvelleCarte()*, qui affiche la question suivante dans le tableau.
- 🌈 *boutonEtape2* et *boutonEtape3*, qui appellent respectivement *chargerEtape2* et *chargerEtape3*, qui gèrent l'affichage des étapes concernées. Ainsi, pour l'étape 2, le tableau apparaît tout de suite. Pour l'étape 3, l'utilisateur doit appuyer sur « Start » pour commencer.
- 🌈 *boutonFin*, qui n'est visible que lorsque l'utilisateur a répondu à la dernière question de la dernière étape et qui déclenche la fonction *terminer()*.
- 🌈 *boutonRejouer*, qui n'est visible que lorsque le bouton précédent *boutonFin* a été cliqué. Il déclenche la fonction *nouvellePartie()*.

De plus, des événements *onmouseover* permettent de modifier le curseur lorsque l'utilisateur survole un élément cliquable comme la flashcard ou les cases du tableau.

Deuxième jeu : Tir spatial

Les principaux éléments HTML qui supportent ce second jeu sont les douze canvas créés :

- 🌈 le canvas *canvas* qui supporte la fusée et l'affichage du score. Ce canvas n'est visible que lorsqu'une partie est en cours. Un événement *onmousemove* y est défini, qui permet de bouger la fusée de façon horizontale : dès que l'utilisateur bouge sa souris, la fusée est effacée et recrée à l'endroit de la nouvelle position utilisateur. Ce déplacement ne s'effectue que sur un axe horizontal, c'est-à-dire qu'on ne modifie que la valeur en x de la fusée. Afin que le curseur de la souris soit au milieu de la fusée, j'ai dû modifier de quelques pixels la position recueillie par l'*event.OffsetX*. Au début, j'utilisais un événement *client*, mais cela occasionnait des décalages lorsque la fenêtre était réduite et qu'on avait scrollé vers la droite.
- 🌈 *canvasLaser*, qui permet de tirer sur les vaisseaux. J'ai dû utiliser un deuxième canvas puisqu'il n'est pas possible de gérer séparément les dessins sur un même canvas. Ainsi, puisque la gestion du laser répondait à un *onclick* et devait disparaître au bout d'un moment, il était plus simple d'avoir un canvas dédié. Ce canvas supporte un événement *onclick* qui tire un rayon laser lorsqu'une partie est en cours et que l'utilisateur clique. Cet événement déclenche la fonction *lancerLaser()* qui fait apparaître le laser à l'endroit où se trouve la fusée. Si le laser ne croise aucun vaisseau, il va jusqu'en haut de la fenêtre. Il s'agit en fait simplement d'un trait blanc dessiné grâce à la méthode *stroke()*, en récupérant la position de la fusée. En tenant compte de la largeur des vaisseaux étrangers, 50 pixels, on peut établir des « fenêtres » dans lesquelles le tir se fait sur un vaisseau. Le trait est donc tracé de la fusée au haut de la fenêtre si on est hors de ces fenêtres, et s'arrête à la hauteur du vaisseau s'il y en a un, en se servant des hauteurs de vaisseaux stockées dans le tableau *tableauPositionsQuestions*. Cette fonction *lancerLaser()* appelle la fonction

destruireVaisseau(), qui prend en paramètre *lieuDuTir + 1*, ce qui correspond au numéro du canvas dans lequel on va devoir opérer. Ainsi, *lieuDuTir* identifie le numéro de la fenêtre correspondant à un vaisseau, de 0 à 9. *destruireVaisseau()* prend donc en paramètre un nombre de 1 à 10, et fait apparaître une explosion et détruit le vaisseau dans le canvas correspondant.

- 🌈 Les différents canvas *canvasProp1* à *canvasProp10* qui vont chacun supporter le vaisseau étranger et la proposition associée. Comme pour le laser, il m'a semblé plus simple et efficace d'attribuer un canvas à chaque vaisseau, plutôt que de devoir tout effacer et redessiner tous les autres à chaque fois qu'un vaisseau était éliminé.

Parmi les objets HTML, on trouve aussi les boutons *boutonCommencer()*, qui déclenche la fonction *initialiserPartie()* et *boutonRejouer()* qui déclenche la fonction *rejouer()*.

Les événements ont été appliqués directement en JavaScript grâce à une fonction *addEventListener*.

Les variables JavaScript contiennent principalement la référence aux objets HTML, ainsi que les différents sons et images nécessaires au jeu, soit le bruit de tir, celui de l'explosion, l'image de fond, l'image de la fusée, celle des vaisseaux et celle de l'explosion. On recense également, comme dans l'autre jeu, des tableaux pour récupérer toutes les questions possibles, pour stocker les dix questions qui seront effectivement posées et un tableau qui permet de gérer le choix des dix questions. Comme précédemment, les tableaux ont deux fois plus de cases que de questions, dans les cases paires, on stocke la proposition, et dans la case impaire qui suit, le booléen associé.

À cela s'ajoute les tableaux *tableauPositionsQuestions*, qui stocke les hauteurs des différents vaisseaux étrangers, et *tableauVaisseauxDisparus*, qui stocke les numéros des vaisseaux qui ont déjà été éliminés.

D'autres variables permettent de faire fonctionner l'algorithme, telles que le *score*, le nombre de vaisseaux à détruire, qui recense le nombre de propositions fausses du tableau des questions à poser, le booléen *fini*, la position de la souris de l'utilisateur ou encore le *timer*, qui permet de modifier le score à chaque seconde.

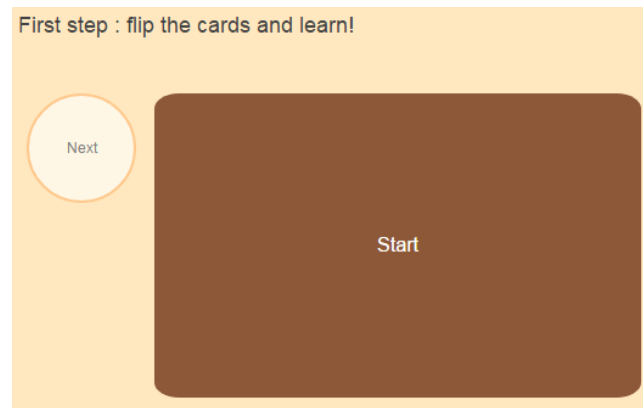
Sessions de jeu

Premier jeu : Flashcards

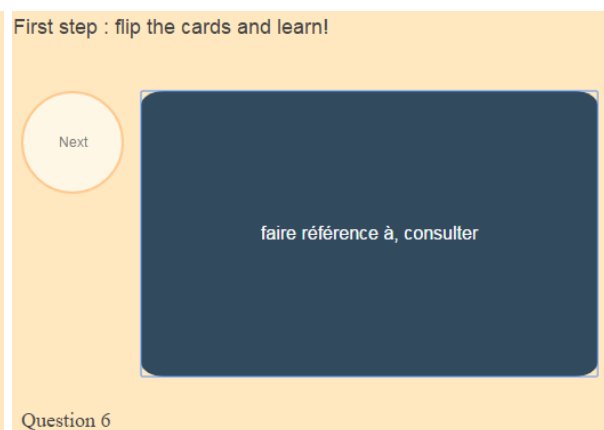
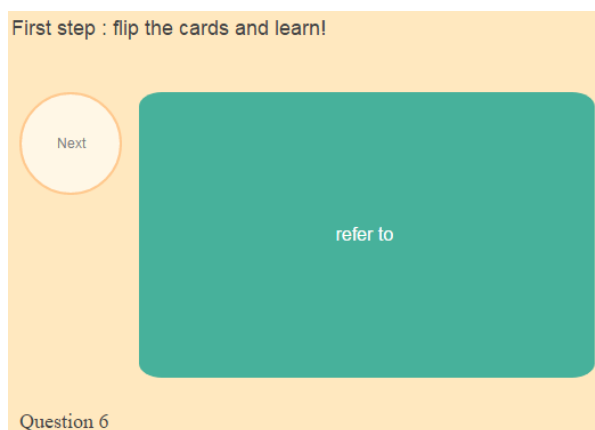
Comme expliqué précédemment, le **premier jeu** est une séquence de flashcards.

C'est une séquence d'apprentissage au cours de laquelle l'utilisateur peut "retourner la carte" en cliquant dessus avant de mettre en correspondance un mot ou une expression anglaise avec l'équivalent en français.

Lorsqu'il commence le jeu, l'utilisateur doit d'abord appuyer sur Start :

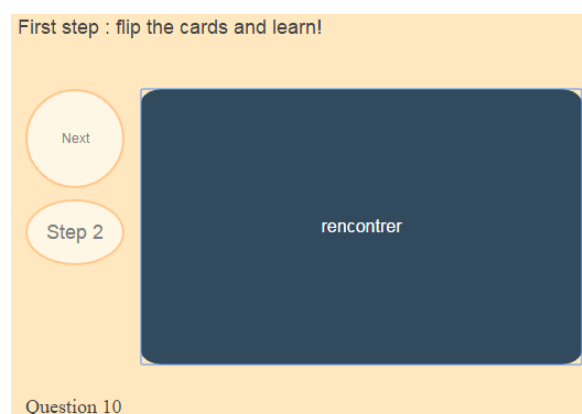


Une première carte s'affiche alors, sur son « recto » : la face sur laquelle le mot est affiché en anglais.



Lorsqu'il clique sur cette carte, elle se retourne. La couleur de la carte change et la traduction s'affiche. Pour passer à la question suivante, il doit cliquer sur le bouton « Next ». Tant qu'il ne le fait pas, il peut cliquer autant de fois qu'il le veut sur la carte.

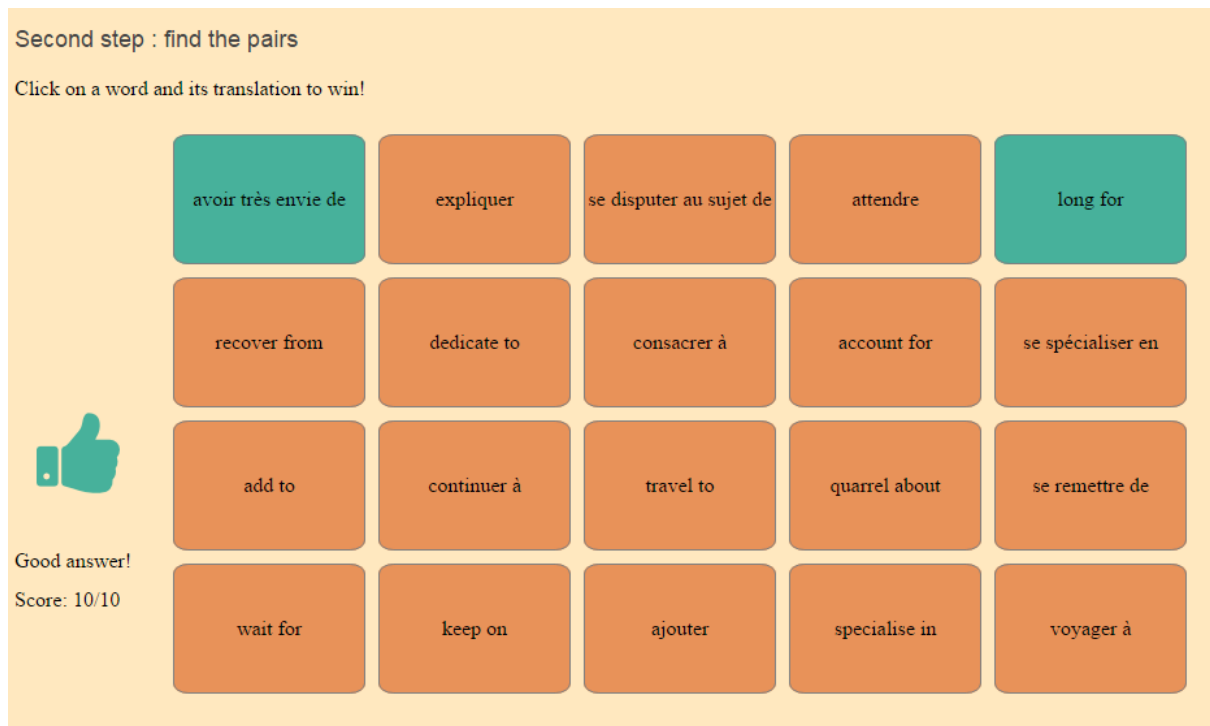
Lorsque l'utilisateur arrive à la dernière question, le bouton « Step 2 » apparaît, et l'utilisateur peut accéder à la deuxième étape :



La **deuxième étape** a pour but d'aider l'utilisateur à mémoriser le contenu des flashcards en lui permettant d'associer des cartes par paire.

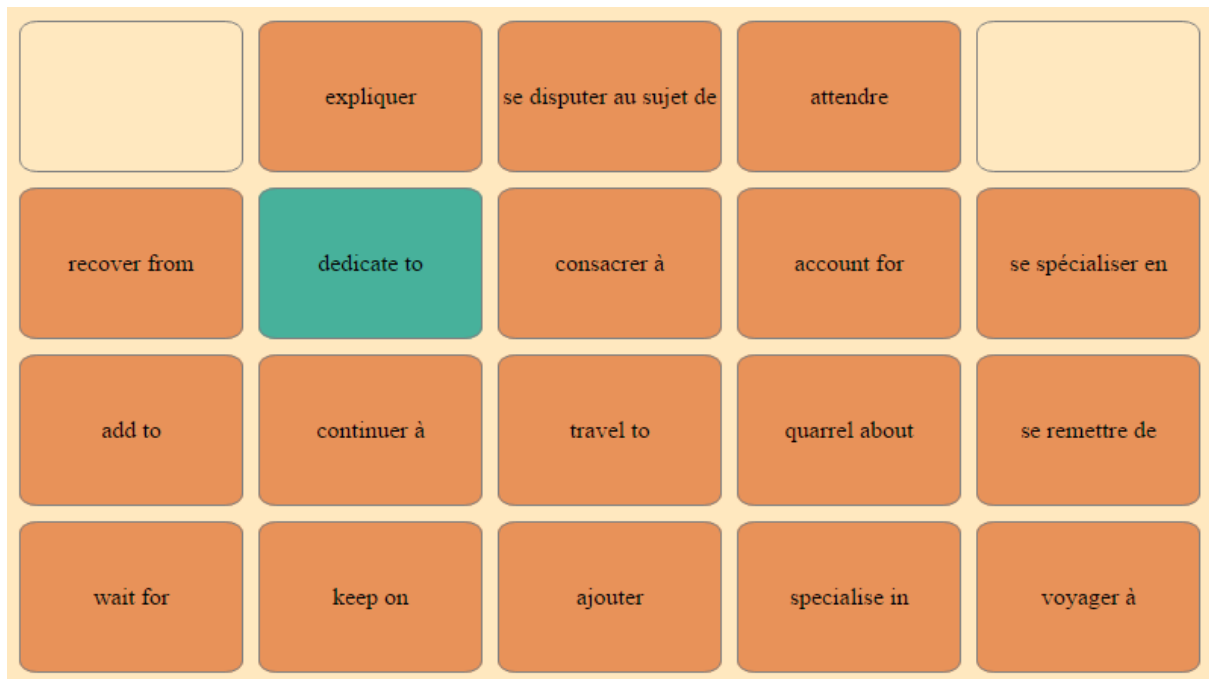
Pour cela, j'ai choisi en accord avec le groupe de transdisciplinaire de réaliser une sorte de jeu de Memory : toutes les cartes sont visibles, et l'utilisateur doit cliquer successivement sur une carte et sur sa traduction.

Si l'association est correcte, une image s'affiche ainsi que « Good answer ! » :

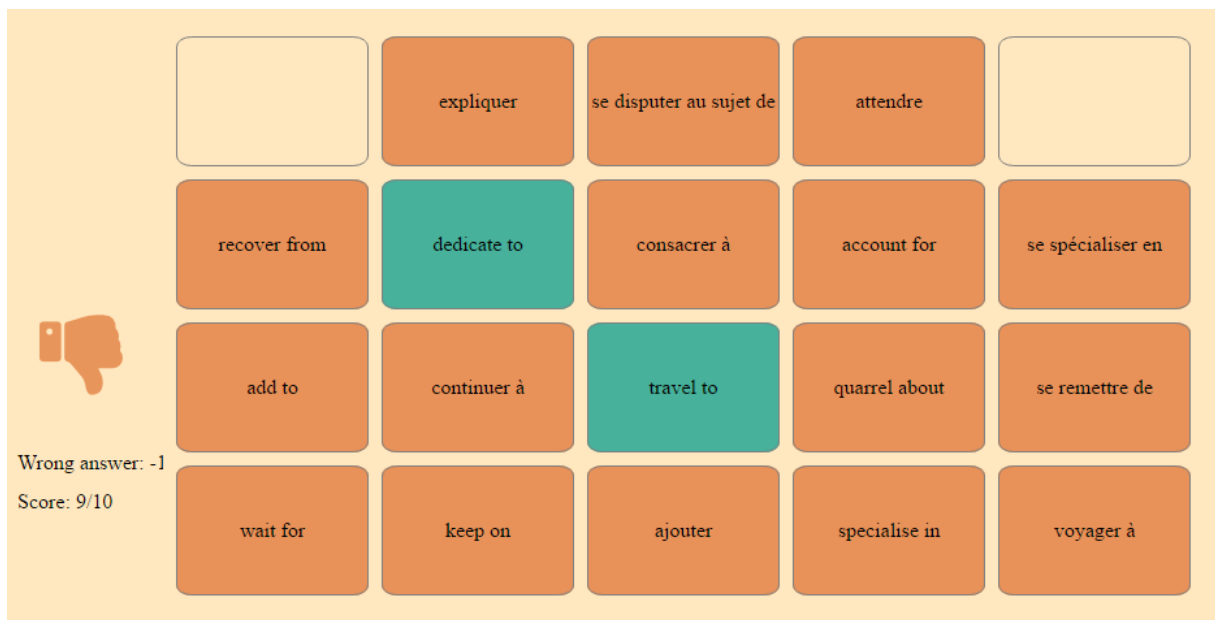


L'image du pouce levé disparaît progressivement, mais la mention « Good answer! » demeure.

L'utilisateur peut alors appuyer sur d'autres cartes, alors que les deux cartes associées disparaissent.



Si l'utilisateur a mal associé les cases, elles se désélectionnent et l'utilisateur perd un point.



Une fois que l'utilisateur a apparié toutes les cartes, le bouton « Step 3 » s'affiche, et l'utilisateur a accès à la troisième et dernière étape :

Step 3

			communicate with	
			être en contact avec	

Good answer!
Score: 10/10

La **troisième étape** a pour but de vérifier la mémorisation en demandant à l'utilisateur de traduire un mot ou une expression. La flashcard de l'étape 1 est affichée, mais elle n'est plus cliquable, et l'utilisateur doit entrer la traduction du mot ou de l'expression qu'elle affiche.

Third exercise : translation

You will now have to write the translation of the words displayed on the cards.

Next

I don't know

travel to

Question 1


Enter your translation:

Pour chaque question, l'utilisateur peut choisir d'indiquer qu'il ne sait pas ou valider sans avoir rien entré, auquel cas il ne gagne pas de point.

You will now have to write the translation of the words displayed on the cards.

Next

I don't know



appartenir

Question 2

Enter your translation:

You gave no answer. The good answer was: appartenir


Your score is: 0/10

L'utilisateur peut également entrer une mauvaise réponse. Dans ce cas, le programme rappelle ce que l'utilisateur a entré et donne la bonne réponse, qui s'affiche également dans la flashcard.

You will now have to write the translation of the words displayed on the cards.

Next

I don't know



expliquer

Question 3

Enter your translation:

You said: compter. That's a wrong answer! The good answer was: expliquer


Your score is: 0/10

Il peut également entrer une bonne réponse :

You will now have to write the translation of the words displayed on the cards.

Next

I don't know



appartenir

Question 4

Enter your translation:

You said: appartenir. Congratulations! That's the good answer!

Your score is: 1/10

À chaque fois que la réponse est affichée, le seul bouton cliquable est le bouton « Next », sur lequel l'utilisateur doit cliquer pour accéder à la question suivante.

Une fois qu'il a répondu à la dernière question, le joueur voit s'afficher la réponse et doit cliquer sur « Quit » pour voir le score :

You will now have to write the translation of the words displayed on the cards.

Next

I don't know

Quit

rencontrer

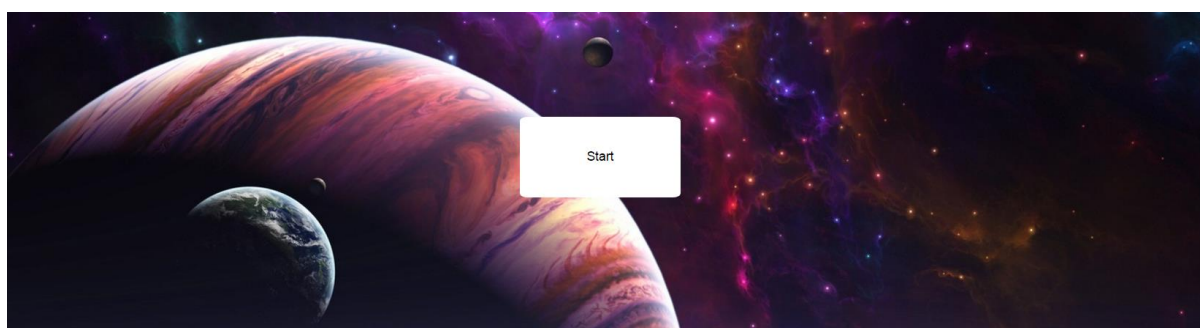
Question 10

End of the game! You scored 10/10 at the pairing game and 4/10 at the translating game.

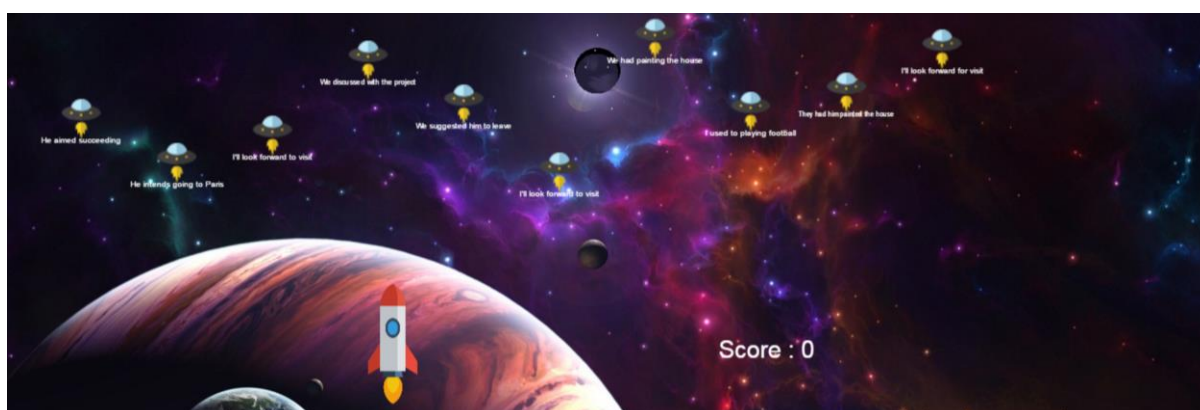
Play Again

Second jeu : Tir spatial

Le but de ce jeu est de déplacer la fusée pour tirer sur les énoncés incorrects. Ainsi, la fusée se déplace horizontalement en suivant la souris.



Pour commencer, l'utilisateur doit appuyer sur « Start ». S'affichent alors la fusée, rouge et blanche, que contrôle l'utilisateur et les dix vaisseaux étrangers qui lui font face, accompagnés chacun d'une proposition rédigée en anglais.

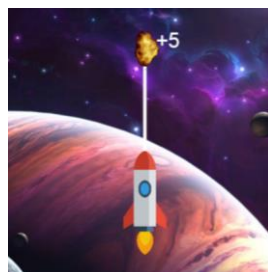




Lorsque l'utilisateur clique, la fusée tire vers le haut. S'il ne croise pas de vaisseau, il ne se passe rien.



Si la fusée se trouve sous un vaisseau étranger, et que celui-ci porte une proposition fausse, il explose, et l'utilisateur gagne cinq points.



Si au contraire le vaisseau porte une proposition grammaticalement correcte, le vaisseau n'explose pas, mais l'utilisateur perd deux points.



Le score ne dépend pas seulement des tirs : il dépend aussi du temps. Ainsi, à chaque seconde, l'utilisateur perd un point. Il doit donc réfléchir le plus justement et le plus vite possible. Cela lui permet d'assurer l'automatisme de ses réponses : s'il joue suffisamment de fois à ce jeu, les tournures grammaticalement correctes lui deviendront comme une seconde nature !

Fonctionnalités

Le tableau ci-dessous résume les différentes fonctionnalités du projet :

FONCTIONNALITES GENERALES	
Les jeux sont codés en JavaScript, HTML et CSS	Fait
Le système peut s'insérer dans la structure du site Eas'Improve	Fait ¹
Le système calcule des scores	Fait
Le système peut renvoyer le score de l'utilisateur	Commencé ²
Le système permet de travailler son anglais en jouant	Fait
Le système fonctionne avec des bases de données externes	Fait
[Facultatif] Le système gère des niveaux de difficulté en fonction du niveau du joueur	Non fait ³
[Facultatif] Le système pose des questions selon les erreurs des sessions précédentes	Non fait ⁴
FONCTIONNALITES DU PREMIER JEU	
Le système a une étape de consultation sans score pour l'apprentissage sous forme de Flashcards	Fait
Le système propose une étape d'appariement des versions française et anglaise (rappel indicé)	Fait
Le système a une étape de rappel non indicé (traduction directe)	Fait
Les trois étapes se font sur la même liste de verbes pour favoriser la mémorisation	Fait

Le système affiche les scores à chaque étape et à la fin	Fait
Le système permet de rejouer	Fait
FONCTIONNALITES DU DEUXIEME JEU	
La fusée se déplace au mouvement de la souris	Fait
La fusée tire des rayons lasers qui sont arrêtés s'ils croisent un vaisseau étranger, qui continuent sinon	Fait
Les vaisseaux portant des mauvaises réponses explosent lorsqu'on tire dessus	Fait
L'utilisateur perd des points en tirant sur des vaisseaux avec les bonnes réponses	Fait
L'utilisateur perd des points en fonction du temps qui passe	Fait

1 : Le code sera fourni au groupe de transdisciplinaire qui pourra simplement l'ajouter à son propre code. Cf infra.

2,3 : la communication entre la base de données des joueurs et mon programme n'étant pas effective, ces aspects sont possibles (le score est calculé, pas renvoyé) mais non mis en place.

4 : Le groupe de projet transdisciplinaire, pendant ne pas avoir le temps d'effectuer les aménagements nécessaires dans leur base de données pour ajouter cette fonctionnalité, a préféré que je ne la code pas.

Bases de données

Les données qui m'ont été transmises sont constituées de deux bases indépendantes, dont la première, « verbes », se présente comme suit :

id_verbe	verbe	prep	trad	lvl
1	account	for	expliquer	3
2	add	to	ajouter	1

Base de données pour le premier jeu

La colonne « lvl » de cette table ne m'a finalement pas servie. Elle devait me permettre d'adapter les questions à poser : pour un utilisateur de niveau 1, seules les propositions de niveau 1, les plus simples, se seraient affichées. Puis, à mesure que le joueur augmente de niveau, plus de propositions auraient dû s'afficher. Cependant, dans l'état actuel du code du projet transdisciplinaire et du mien, je ne récupère pas d'informations sur l'utilisateur qui joue, et je ne peux donc pas connaître son niveau.

La seconde table, « jeuTir », n'a que deux colonnes : l'une présente une expression en anglais, la seconde contient un booléen associé à cette expression, selon qu'elle est grammaticalement correcte ou non.

Explications techniques

Le projet se compose de deux dossiers indépendants, un pour chaque jeu.

🌂 Le premier dossier se compose de :

- 👤 *code.html* : le fichier HTML à lancer pour démarrer le jeu. Ce code fait référence au fichier CSS et au fichier JS. Il définit les objets HTML sur lesquels se base le code en JavaScript.
- 👤 *CSSpartie1.css*
- 👤 *source.js* : le fichier qui contient à proprement parler l'algorithme du jeu, rédigé en JavaScript, avec par endroits des instructions qui font appel à jQuery.
- 👤 Les deux images *goodAnswer.png* et *wrongAnswer.png*, qui s'affichent lorsque l'utilisateur entre une réponse.

🌂 Selon le même schéma, le second dossier se compose de :

- 👤 *jeu2.html*
- 👤 *stylePartie2.css*
- 👤 *jeu.js*
- 👤 Les images *rocket.png*, *vaisseauAlien.png* et *giphy.gif*, qui sont les représentations de respectivement la fusée du joueur, les vaisseaux étrangers, et l'explosion d'un vaisseau, ainsi que *cosmos.jpg* qui est l'image de fond.
- 👤 Les sons *sonExplosion.mp3* et *sonTir.mp3*, déclenchés lors de l'explosion d'un vaisseau ennemi ou d'un tir de la fusée.

Chargement des données

Afin de m'assurer que les données seraient bien chargées lorsque l'utilisateur appuie sur « Start », j'ai désactivé ce bouton au chargement de la page, et il n'est réactivé que dans la *callback* de la fonction *ajaxGet*. Cependant, malgré cette précaution, il arrive que les données ne chargent pas assez vite, et que la flashcard affiche *undefined* au clic sur Start. Parfois, cliquer sur la carte suffit pour que les données s'affichent normalement, et parfois, il faut recharger la page.

Réutilisation des objets HTML

Dans le premier jeu, un certain nombre d'objets servent dans plusieurs étapes. Par exemple, le bouton « Start » est en réalité *boutonCarte*, c'est-à-dire la flashcard, qui affiche Start au

début de la partie et qui affiche les questions et réponses après le premier clic. Dans la troisième partie, c'est toujours la même flashcard qui s'affiche, mais elle n'est cette fois plus cliquable, et ne se retourne que lorsque l'utilisateur valide une réponse, de façon à afficher la réponse.

Cette réutilisation des objets permet de factoriser certaines sections de code et de limiter le nombre d'objets qui ne servent qu'une fois et doivent être cachés le reste du temps. Par exemple, la fonction *verifierReponse()*, qui sert dans la partie 3 du premier jeu, réutilise la fonction de retournement de la partie 1.

Utilisation de canvas comme layers pour l'affichage de la fusée, des vaisseaux et du laser

L'affichage dans le second jeu se compose de 12 canvas superposés, qui permettent de créer des « couches » de dessin avec un ordre de priorité. Pour ce faire, il m'a fallu attribuer à chaque canvas un positionnement relatif, puis à la <div> qui contient tous les canvas un positionnement absolu. Enfin, pour gérer les positionnements en premier ou en arrière-plan, j'ai utilisé la propriété *z-index*.

Problèmes rencontrés

J'ai conscience que mon code pourrait être optimisé, mais, pour l'instant, je me concentre sur le fait que les jeux fonctionnent et sur le seul JavaScript.

J'ai donc conscience que de nombreuses parties de mon code pourraient peut-être être simplifiées, raccourcies ou optimisées, par exemple grâce à jQuery, mais j'ai préféré gardé le travail que j'avais déjà effectué et qui était fonctionnel, et me concentrer sur l'ajout de fonctionnalités comme la possibilité de rejouer, plutôt que sur la reprise de tout mon code.

La façon de reconnaître les réponses pose un problème également rencontré par d'autres sites d'apprentissage de langues : lorsque l'on fait une étape de rappel non indicée, au cours de laquelle l'utilisateur doit entrer au clavier une traduction, il est possible que sa réponse soit refusée alors qu'elle est en substance juste. Par exemple, si l'utilisateur avance comme traduction à « crash into », « heurter, percuter », le programme le lui comptabilisera comme une mauvaise réponse. Or, la bonne réponse est « percuter, heurter ».

C'est un problème qu'on rencontre également sur des sites comme Quizzoodle, auquel on pourrait remédier en admettant plusieurs réponses possibles à une même proposition, ou en ne tenant pas compte de certains caractères comme un espace maladroît à la fin d'un mot, en mettant en place un système qui admettrait comme juste une réponse à partir du moment où seulement une lettre change, en cas de faute de frappe... Il faudrait peut-être aussi revoir légèrement la base de données, dans la mesure où la préposition associée au verbe est parfois présente en français, et parfois non. Par exemple, « prepare for » est traduit dans la base de données par « préparer », alors qu'une traduction plus complète pourrait être « préparer à » ou « préparer pour ».

Par ailleurs, les synonymes peuvent poser problème au vu de la façon dont j'ai codé la vérification des appariements pour la deuxième partie du premier jeu. Ainsi, le programme parcourt le tableau des questions et réponses pour trouver le texte qui est dans la case cliquée. Mais s'il y a des synonymes, deux cartes peuvent être identiques, et l'appariement se fera avec la dernière carte correspondante rencontrée dans le tableau. Par exemple, *collide with* et *crash into* se traduisent tout deux par *percuter, heurter*. Si les deux paires sont présentes dans la partie, l'utilisateur pourrait cliquer sur une des cartes *percuter, heurter*, et, en cherchant dans le tableau, le programme attendrait l'une des expressions pour la deuxième case cliquée, et pas n'importe laquelle des deux. C'est un défaut que je n'ai pas eu le temps de changer.

Tests

Les différentes fonctionnalités ont été testées **au fur et à mesure** qu'elles ont été codées. De plus, des tests de fonctionnement globaux ont été réalisés à la fin du projet. Le code des fonctions qui permettent de rejouer ayant été réalisé à la fin, pour les deux jeux, il a révélé de nombreux problèmes d'affichage et de réinitialisation des variables, qui m'ont permis de tester à nouveau toutes les fonctionnalités.

Des tests utilisateurs avaient été prévus afin d'améliorer certains détails, mais je n'ai pas eu le temps de les mettre en œuvre. Cependant, des commentaires et retours notamment des élèves du projet transdisciplinaire et d'autres élèves de l'ENSC m'ont permis de revoir certains détails, surtout au niveau de l'affichage et des couleurs.

Intégration et reprise du code

Intégration de mon code dans le site Eas'Improve

Mon code ayant été rédigé en JavaScript, un langage web standard, à partir d'une base HTML-CSS, il n'y aura pas de problème à l'intégrer dans le site créé par le groupe de transdisciplinaire. En effet, il leur suffira de d'intégrer mes éléments HTML dans une balise `<div>` de leur site, sur la page sur laquelle devront être insérés les jeux. Il leur faudra aussi créer des références dans leur fichier HTML à mes fichiers en JavaScript, et d'ajouter mes éléments de CSS aux leurs.

Cependant, il faudra faire attention à d'éventuels conflits de nommage, si jamais le code du site comprenait des éléments nommés de la même façon que dans un de mes jeux. De même, si les deux jeux sont affichés sur la même page, il faudra renommer certains éléments afin qu'ils ne soient pas confondus, notamment *boutonRejouer*, qui apparaît dans les deux fichiers HTML.

Reprise possible du code, sécurité

Le code fourni est fonctionnel sur les bases de données qui ont été fournies à l'API que j'utilise pour les récupérer. Si les bases de données changent de nom, d'emplacement ou de structure, mon code devra être adapté.

De plus, les bases de données sur lesquelles j'ai travaillé sont temporaires, et à un emplacement temporaire peu sécurisé. Si le site est mis en ligne, il faudra gérer l'API plus finement afin qu'il soit par exemple impossible de modifier la base de données sans authentification.