
# Projet Tux Letter Game

08.12.2021

Begimay KONUSHBAEVA

Rayan GUERBAA

# Introduction

Nous devons réaliser au cours de ce projet un mini-jeu éducatif dans un environnement 3D. Ce jeu permet d'apprendre l'orthographe des mots de façon ludique, en faisant déplacer un petit personnage pour assembler dans l'ordre des lettres dans un temps imparti.

Pour ce projet, nous devions utiliser les connaissances acquises en langage XML et Java pour concevoir le jeu.

## Objectifs

Dans un premier temps, nous devons structurer les données du jeu à travers le langage XML. Nous avons alors écrit différents fichiers :

- un dictionnaire (XML, XSD, XSL) contenant l'ensemble des mots.
- un profil (XML, XSD, XSL) contenant les informations personnelles du joueur ainsi que ses parties jouées.
- un environnement (XML) prédéfini pour le jeu.

Ensuite, nous devons à travers le langage Java concevoir le jeu, ainsi que les différents menus affichés au joueur. Nous avons alors programmé à l'aide des fichiers fournis :

- les menus d'affichage permettant de passer correctement du menu principal au menu joueur et inversement.
- la création de nouveaux profils.
- le chargement de profils existant.
- le jeu en lui-même.
- la sauvegarde de profil en format XML et HTML

## Conception du jeu

Pour la conception du jeu, nous avons repris l'ensemble des fichiers fournis et suivi au maximum les consignes des différentes parties du projet. Toutefois, pour la partie programmation Java du jeu, nous avons effectué quelques changements sur certaines fonctionnalités du jeu. Par exemple,

- l'assemblage des mots pour gagner le jeu.
- le déplacement des lettres, qui s'effectue sur la tête du personnage.
- l'affichage des scores.
- l'ajout de nombreux affichages dans l'environnement et dans le terminal.
- le niveau de difficulté des mots, représentant dans notre cas la longueur du mot associé à un temps de partie.
- la possibilité de retrouver des mots avec accent.
- la génération d'un fichier profil XML, HTML et l'affichage des informations du profil.

Ces fonctionnalités seront détaillées ci-après.

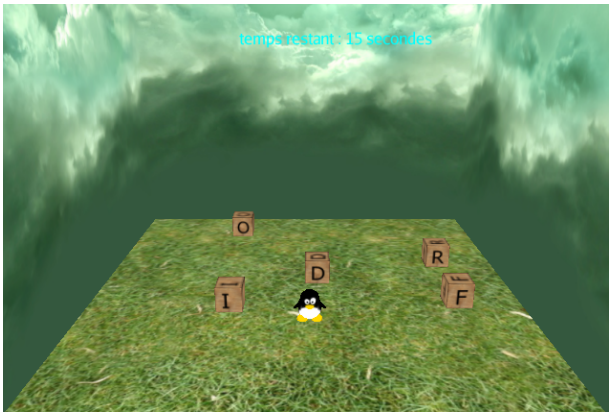
# Règles du jeu

Les règles du jeu que nous avons choisi d'appliquer sont très similaires au jeu initialement proposé, qui sont de trouver dans l'ordre les lettres d'un mot dans un temps imparti.

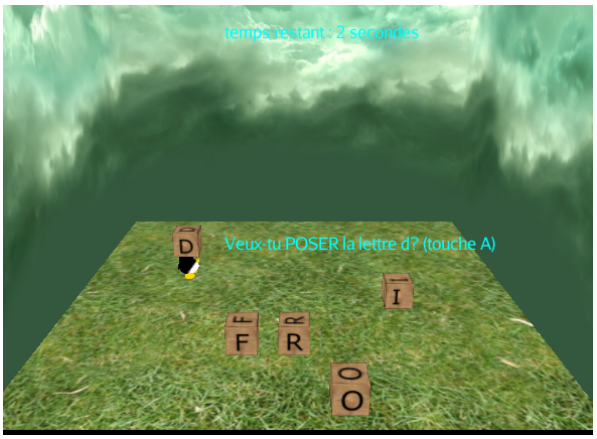
Mais au lieu de simplement chercher la bonne lettre dans l'environnement, le joueur doit également rassembler une par une les lettres du mot, en les plaçant correctement dans l'ordre dans l'environnement.
Les lettres seront disposées aléatoirement dans l'environnement tux de la partie, le joueur doit donc rassembler les lettres du mot en temps imparti. Il devra donc récupérer chaque lettre, la déplacer, et la déposer dans le bon ordre pour reconstituer le mot.
La position des lettres sera analysée selon l'axe horizontal  tout au long de la partie pour vérifier si le joueur a réussi à assembler le mot dans l'ordre ou non (dépend de l'axe X, ne dépend donc pas de l'axe Z). Pour exécuter une partie, il suffit de RUN le fichier LanceurDeJeu.java.

Voici un exemple de début de partie :

Les lettres sont placées aléatoirement dans l'environnement.




Ici, le joueur va récupérer les lettres et les placer dans l'ordre selon l'axe horizontal.




Dans ce cas là, le mot à rechercher est "froid", pour que le joueur finisse sa partie, il lui suffit de placer la lettre "D" qu'il porte sur la tête à la droite du "I".

# Menu Principal

Au lancement du Jeu l'on propose à l'utilisateur 3 options suivantes :

1. Charger un profil d'un joueur existant
2. Créer un nouveau joueur
3. Ajouter un mot au dictionnaire
4. Consulter les scores d'un profil
5. Quitter le jeu

## Création et chargement des profils

Dans notre implémentation du Jeu il est possible d'instancier un profil de deux façons différentes:

- En connaissant le nom du joueur existant. Dans ce cas, toutes les données se trouvant dans le fichier XML de joueur sont recopiées dans l'objet Java de type Profil.

- En tapant dans le menu nouveau joueur, le nom et la date de naissance du joueur. Dans ce cas, ces données sont stockées directement dans l'objet Java de type Profil.

De plus, Les profils se trouvent dans le dossier xml Joueurs du dossier Data.

## Charger un profil existant

Dans le cas où l'utilisateur choisit la première option, l'application lui demandera de saisir le nom du joueur existant.
Si le profil existe, donc figure parmi la liste des fichiers XML, on fait appel au constructeur spécifique pour récupérer son profil.
Si le profil n'existe pas, donc ne figure pas parmi la liste des fichiers XML joueurs, on renvoie gentiment l'utilisateur vers le menu principal en lui suggérant de recommencer ou bien de créer un nouveau profil.

Une fois que le profil a été trouvé, les informations concernant le profil sont lues depuis le fichier du joueur. Ces données, dont le nom, la date d'anniversaire et les parties jouées, sont ensuite stockées dans le profil actif (celui qui joue). Ensuite, lorsque le joueur quitte le jeu ou effectue des parties, ces informations sont converties en éléments du document DOM et enregistré dans le dossier xml Joueurs du dossier Data.

Dans notre implémentation du jeu nous avons décidé de stocker chaque profil indépendamment dans un fichier sous forme *nom_de_joueur.xml*.

## Créer un nouveau joueur

Dans le cas où l'utilisateur choisit la seconde option, l'application lui demandera de saisir son nom ou pseudonyme et sa date d'anniversaire. Dans ce cas là, on fait l'appel au deuxième constructeur d'un objet de type Profil.
Les informations saisies par l'utilisateur sont directement enregistrées dans le profil.

## Ajouter un mot au dictionnaire

Dans ce cas là, on demande à l'utilisateur de saisir au clavier un mot qu'il souhaite ajouter au dictionnaire, ensuite on demande le niveau de ce mot.

Pour ajouter des nouveaux mots au dictionnaire on utilise un objet de type EditeurDico. Cette classe possède en attributs, une liste pour chaque niveau de difficulté des mots ainsi qu' un document DOM. Suivant les consignes du projet nous avons défini deux méthodes : la lecture de document et l'écriture du document lu.

L'application va donc d'abord lire le fichier xml avec le dictionnaire existant et le transforme en une instance de Java. Tous les mots du dictionnaire se retrouvent dans les listes de l'instance Java de EditeurDico.

Puis on prend en entrée du constructeur, le mot a ajouter dans le dictionnaire que l'on va ajouter dans une des listes de instance Java en fonction de son niveau. Le document DOM est par la suite réécrit et transformé en fichier XML à l'aide des méthodes fournies.

## Consulter un profil

Si l'utilisateur choisit cette option, l'application lui demande de saisir le nom du profil qu'il veut consulter. Ce nom est ensuite vérifié. Si le profil recherché n'existe pas, on renvoie l'utilisateur vers le menu Principal.

Dans le cas contraire, le fichier XML du joueur avec ce nom est lu et stocké dans la mémoire d'une instance Java. La chaîne de caractères contenant des informations sur le profil est donc affichée à l'écran par groupe de 10 lignes pour tout faire rentrer dans le menu.

Pour voir cette fonctionnalité, vous pouvez rechercher le profil "rayan" ou "begimay" pour afficher les quelques parties en une seule fois, ou bien le profil "toto" pour voir l'autre affichage.

# Menu Joueur

Dès que le profil a été trouvé ou créé. On propose à l'utilisateur les options suivantes :

1. Commencer une nouvelle partie
2. Affichage score profil
3. Sortir du jeu
4. Quittez le jeu

## Commencer une nouvelle partie

Lorsque l'on commence une nouvelle partie, on demande au joueur de choisir le niveau de difficulté du mot souhaité. On instancie ensuite une partie vide avec un constructeur par défaut que l'on va initialiser au fur et à mesure avec le niveau choisi, et le mot correspondant à ce niveau sélectionné aléatoirement par le programme.
Avant de commencer à jouer, on initialise un chronomètre à un temps correspondant à la difficulté du mot (niveau : 1 - 2 = 20s, 3 = 30s, 4 = 60s, 5 = 120s).

Les lettres de ce mot sont placées aléatoirement dans la room. Pour faciliter le jeu, on affiche le mot à assembler pendant 2 secondes et ensuite on lance le jeu.

Le jeu ouvre une room instancié au préalable avec les dimensions et le contenu visuel se trouvant dans le fichier plateau.xml. On peut donc changer l'apparence de la room ou ses dimensions en modifiant le fichier plateau.xml. Le personnage tux en début du jeu est placé au milieu de la room. Dans notre implémentation nous avons défini les règles ainsi que

- Les lettres du mot à assembler apparaissent dans un ordre aléatoire
- Le personnage du jeu se déplace et peut également déplacer les lettres

On déplace le personnage tux avec la fonction déplace définie pour tous les objets de type Tux. Pour déplacer les lettres on met la position de la lettre que l'on veut déplacer, à la position de tux à chaque déplacement du personnage.

Quand le personnage tux entre en collision avec une lettre, on propose à l'utilisateur de prendre cette lettre à l'aide des touches (E). Quand le personnage tux prend la lettre, on propose à l'utilisateur de poser la lettre quand il le souhaite en cliquant sur la touche (A).

Ensuite avec chaque déplacement des lettres on calcule l'ordre des lettres du mot assemblé pour décider si le joueur a gagné et dans le cas contraire calculer le pourcentage de la correspondance du mot assemblé au mot de dictionnaire.

A la fin de la partie on ajoute la partie jouée à la liste des parties de profil actif et on sauvegarde les données, dont les informations personnels et la liste des parties jouées, dans un fichier (nom du joueur).xml.

## Sauvegarde des parties

Pour sauvegarder les parties effectuées par un joueur, il suffit de créer ou charger un profil existant, puis de lancer une partie. Lorsque une partie se termine, elle se sauvegarde automatiquement dans le profil du fichier XML du joueur et est transformée en html. De plus, on peut voir les informations de la partie dans l'environnement du jeu ainsi que dans le terminal.

## Choix d'implémentation

En plus des fonctionnalités proposées dans le menu principal et le menu jeu, nous en avions modifié certaines.

Nous avons considéré que la fonctionnalité chargée une partie existante n'était pas pertinente dans ce jeu. En effet, les parties sont très rapides et faciles à faire, nous ne voyons pas l'intérêt de coder cette fonctionnalité dans ce type de jeu. Nous l'avons donc retiré du menu de joueur.

Ensuite, nous avons ajouté d'autres images pour les objets lettres afin de pouvoir jouer avec des mots à accent ou contenant une "ç". Cela donne plus de possibilités au joueur, et donc lui permet d'apprendre plus de mots.

# Description des différents fichiers du projet

## XHTML

Nous avons créé un html qui affiche sous forme de tableau les scores fournis.

## XML

Les fichiers XML de notre projet se trouvent dans le dossier Data/xml du dossier src.
Les deux fichiers demandés dans cette étape du projet sont plateau.xml et dico.xml

Ces deux fichiers respectent le schéma de l'arborescence du document et donc l'encapsulation de tous les éléments.

- Dictionnaire
  Un dictionnaire contient un nombre indéfini de mots. Chaque mot de ce dictionnaire possède un attribut niveau.
- Plateau
  Le plateau contient les dimensions utilisées dans la suite pour définir l'environnement graphique et les fichier avec le contenu visuel de cet environnement.

## XML Schema

Les XML schemas de notre projet se trouvent dans le dossier Data/xsd du dossier src.

## XSLT

Pour cette partie, nous récupérons les informations du profil d'un joueur, puis nous les transformons dans le format HTML après avoir sauvegarder un profil. à chaque fois que l'on joue une partie avec un profil, le fichier HTML est généré et mis à jour.

On peut voir ci-dessous 2 exemples de profils générés en HTML.

**Profil du Joueur**

**Informations**

## Nom joueur : begimay

## Anniversaire : 2002-03-05

### Parties jouées

| Date | Mot | Niveau | Trouvé | Temps |
|---|---|---|---|---|
| 12-12-2021 | coque | 4 | 40 % | Temps maximum |
| 12-12-2021 | ordinateur | 5 | 20 % | Temps maximum |
| 12-12-2021 | froid | 2 | 40 % | Temps maximum |
| 12-12-2021 | noir | 1 | 100 % | 12 |

**Profil du Joueur**

**Informations**

## Nom joueur : rayan

## Anniversaire : 2001-05-01

### Parties jouées

| Date | Mot | Niveau | Trouvé | Temps |
|---|---|---|---|---|
| 12-12-2021 | crayon | 3 | 16 % | Temps maximum |
| 12-12-2021 | crayon | 3 | 50 % | Temps maximum |
| 12-12-2021 | coque | 4 | 60 % | Temps maximum |
| 12-12-2021 | cle | 1 | 100 % | 18 |

## Parsing DOM et SAX

DOM et SAX sont deux parseurs que nous devions utiliser lors de ce projet pour lire et écrire dans un document. Nous avons donc implémenté des méthodes permettant de parser en utilisant DOM et SAX nos fichiers profil.xml et dico.xml.

Cependant, nous avons principalement utilisé le parseur DOM en raison de son utilisation simple et rapide par rapport au parseur SAX. Toutefois, nous avons tout de même utilisé le parseur SAX pour montrer que nous savions également l'utiliser.

En effet, pour le fichier dico.xml, nous avons codé les 2 méthodes de parsing DOM et SAX pour récupérer les mots du dictionnaire afin de les ajouter dans les listes java selon leur niveau de difficulté.

Le parsing SAX est utilisé une seule fois, lors du lancement du jeu, lorsque l'on RUN l'application.

Alors que le parsing DOM, est utilisé pour mettre à jour le fichier dico.xml lorsque l'on ajoute un nouveau mot à travers le menu.

Nous utilisons aussi le parseur DOM pour le fichier profil, lorsque l'on souhaite récupérer les informations d'un profil existant en document xml, mais également pour créer un nouveau document xml lors de la création d'un profil, ou bien pour sauvegarder une partie.

## Conclusion

Pour conclure sur ce projet, nous avons beaucoup apprécié passer du temps dessus, cela nous a grandement aidé à assimiler les différentes notions de XML et Java et permis de les appliquer concrètement.

Nous sommes plutôt satisfait de l'avancée de l'avancée de notre travail, le jeu est fonctionnel et correspond à nos attentes, la structure et les affichages de l'ensemble restent cohérents.

Toutefois, si nous avions eu un peu plus de temps, nous aurions voulu aller un plus loin dans ce projet, en implémentant d'autres fonctionnalités. Comme par exemple :

En utilisant les affichages des scores de manière plus approfondie pour qu'on puisse voir les scores des meilleurs joueurs par tranche de niveau et mots.
En améliorant les affichages des menus et des parties pour les faire ressembler un peu plus a un vrai jeu.
En ajoutant des sons et musiques pour immerger le joueur dans l'ambiance du jeu.