Projet
Tux Letter Game

6 DECEMBRE 2022

Réalisé par :

-OUCHANE Souleymen

-DAMESSI Samuel

Sommaire

Introduction

Dans le cadre de l'UE « Formalisation des Données – Technologies », nous avons été introduit aux technologies XML puis acquis une maitrise relative de ces derniers. Ces technologies nous permettent entre autre de formaliser des données et de les stocker sous forme de document xml, de vérifier la conformité de fichier existant ou devant être créé sous un format prédéfini avec un schéma xml (xsd) et pour finir de pouvoir faire des mises en forme différentes comme passer un fichier xml en fichier html ou autres format avec des feuilles de style xslt.

I - Conception Tux

Tux Letter Game comme son nom l'indique est un jeu de lettre ou l'utilisateur une fois rentré dans le jeu doit retrouver les différentes lettres du mot de la partie en cours. Tux a besoin pour fonctionner de certaines informations qu'il faut obligatoirement stocker dans une source externe car l'écrire en dure dans le programme est une option limitant le choix des mots ; néanmoins les informations ne sont pas assez conséquentes pour qu'on ait forcément besoin de faire appel à un système de gestion de base de données. Les technologies XML s'avèrent être une option idéale pour stocker des bases de donné légère que représente les différentes informations et paramètres du jeu.
Les informations dont Tux a besoin sont divisé en trois grandes catégories

*le profil du joueur

-un fichier xml contenant un profil

-un fichier xsd pour valider tout modification ou création de nouveau profil

-le dictionnaire de mot

-un fichier xml contenant un dictionnaire

-un fichier dico.xsd pour valider tout modification ou création de nouveau profil

-les paramètres de l'environnement du jeu

-un fichier plateau.xml contenant les informations de la scène du jeu

II - Utilisations des technologies XML pour le traitement des données

Profil (Cas d'Utilisation d'un parseur DOM) :

Un profil est propre a chaque joueur ; il peut être déjà existant ou créer au cours du jeu(si il s'agit d'un nouveau joueur).
Un profil joueur contient les informations suivante :

-le nom

-l'avatar

-l'anniversaire

-ses parties : pour chaque partie, on la date ou elle a été joué, un attribut trouvé qui nous indique le nombre de lettre l'utilisateur a trouvé au cours de la cette partie, le temps mis ainsi que le mot et son niveau de difficulté.

Tout cela est présenté sous un format comme le suivant stocké dans un fichier xml.

```xml
<profil xmlns="http://myGame/tux"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://myGame/tux ../xsd/profil.xsd">
    <nom>alban</nom>
    <avatar>conan.jpg</avatar>
    <anniversaire>2006-05-08</anniversaire>
    <parties>
        <partie date="2013-12-10">
            <temps>20.0</temps>
            <mot niveau="1">cle</mot>
        </partie>
        <partie date="2013-12-11" trouvé="75%">
            <mot niveau="2">indice</mot>
        </partie>
        <partie date="2013-12-11">
            <temps>20.0</temps>
            <mot niveau="2">indice</mot>
        </partie>
    </parties>
</profil>
```

Pour que tous les profils aient le même format et que le programme puisse le lire nous faisons appel a un XML schema (xsd) que nous avons défini dans le fichier « profil.xsd » qui se trouve dans le sous répertoire « xsd » du répertoire « data ». Ce schema vérifie que pour tout profil existant ou allant être crée toutes les informations requises sont bien présentent et dans le format indiqué.

Dans le jeu lors du lancement un profil par défaut est créé puis l'utilisateur a le choix de charger un profil existant ou créer un nouveau joueur. Si l'utilisateur choisi de charger un profil existant il devra fournir son nom. Et si ce nom est associé a un profil stocker dans l'un des fichiers du sous-répertoire xml du répertoire xml alors ; on fait appel a un

parseur DOM pour aller récupérer ce profil ainsi que toutes ses informations pour venir initialiser le profil joueur qui sera utiliser dans le jeu.
Si l'utilisateur choisi par contre de créer un nouveau joueur, il devra entrer son nom, alors le programme lui créera un profil ainsi qu'une partie et la fin de la partie les informations du le programme créer document xml ayant une structure conforme au schéma défini dans profil.xsd et y enregistrera les informations du joueur ainsi que sa ou ses parties

Dictionnaire

Le dictionnaire représente l'ensemble des mots pouvant être utilisé dans le jeu et est contenu dans le fichier dico.xml du sous répertoire « xml » du répertoire « data ». Ce dictionnaire est structuré comme suit :

```
<dico
    xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
    xmlns='http://myGame/tux'
    xsi:schemaLocation='http://myGame/tux ../../../build/classes/data/xsd/dico.xsd'>

    <niveaux>
        <niveau niv="1">
            <mot>salut</mot>
            <mot>rue</mot>
            <mot>trou</mot>
            <mot>toit</mot>
        </niveau>

        <niveau>

        <niveau>

        <niveau>

        <niveau>
    </niveaux>

</dico>
```

Cette structure est vérifié et validé par un XML schema défini dans le fichier « dico.xsd » qui se trouve dans le sous répertoire « xsd » du répertoire « data ». Ce schéma permet de s'assurer qu'au moment de l'ajout d'un nouveau mot dans le dictionnaire ou dans le cas où on voudrait créer un nouveau dictionnaire, cette structure soit gardée et que le dictionnaire soit toujours interprétable par le programme.

A chaque fois que le jeu est lancé on initialise un dictionnaire qui grâce a un parseur va parcourir le dictionnaire et dans chaque niveau récupérer les mots du pour les affecter dans des listes en mémoire du programme correspondantes à chacun de ses niveaux.
Nous faisons appel a la lecture par parseur SAX dans la fonction getNiveau_Mot() de la classe Jeu qui demande au joueur d'entrée un niveau de difficulté.

L'environnement

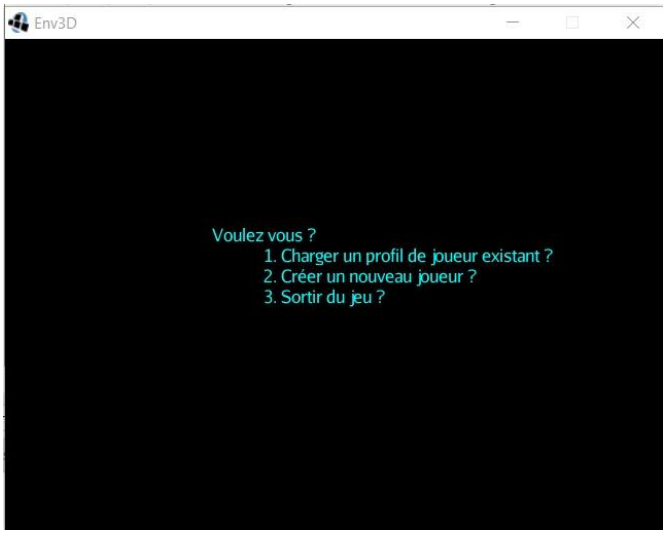
Pour mettre en place l'environnement du jeu avant qu'il ne commence on a besoin de certains paramètre tel que les texture la profondeur, la hauteur, la largeur….Tous ces paramètres sont contenu dans le plateau.xml et formaté de la façon suivante

```
<plateau>
    <dimensions>
        <height>60</height>
        <width>100</width>
        <depth>100</depth>
    </dimensions>

    <mapping>
        <textureBottom>textures/floor/grass2.png</textureBottom>
        <textureEast>textures/skybox/interstellar/east.png</textureEast>
        <textureWest>textures/skybox/interstellar/west.png</textureWest>
        <textureNorth>textures/skybox/interstellar/north.png</textureNorth>
    </mapping>

    <pion>une url</pion>
</plateau>
```

Ce plateau est validé par un XML schema défini dans le fichier plateau.xsd du sous répertoire « xsd » du répertoire « xsd » pour s'assurer que pour chaque plateau on ait toutes ces informations.
Nous utilisons un parseur DOM pour récupérer ses valeurs et initialiser les attributs de la classe Room.

III - Fonctionnalités opérationnelles

Une fois le jeu lancé, l'utilisateur à accès au menu suivant



En en appuyant sur « 1 »(cela peut être la touche 1 du pavé numérique ou celui du pavé alphanumérique) l'utilisateur a accès a la fonction suivante

Chargement d'un profil existant

Cette fonctionnalité demande au joueur d'entré son puis vérifie si le fichier xml contenant le profil associé a ce nom existe et si c'est le cas elle fait appel a un parseur DOM pour récupérer tout le profil dans un objet profil en mémoire du programme.

Chargement d'une partie existante

Une fois que le profil existant est récupéré choix de charger une partie précédente ou une lancer une nouvelle partie. Dans notre projet nous avons laissé le choix de la partie a charger au programme, le joueur n'a pas le choix. Dans le cas d'un nouveau joueur cette fonctionnalité n'est pas utile et essayé de l'utiliser renvoie au menu précèdent.

Chargement aléatoire d'un mot provenant du fichier dico.xml

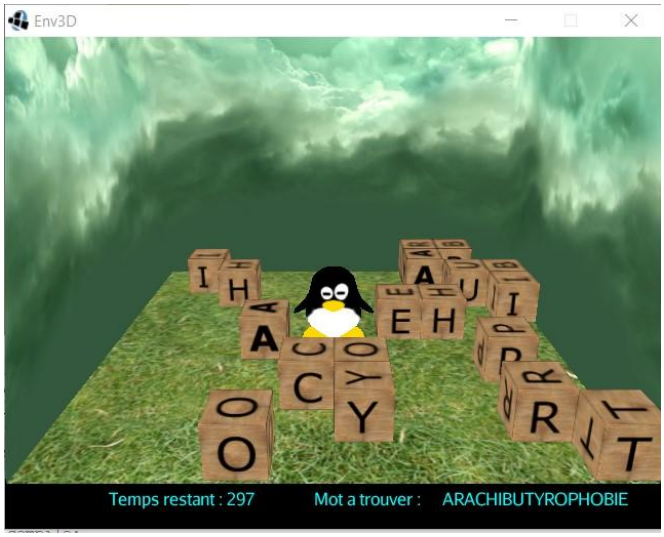
Apres le parsing sax qui nous permet de récupérer les mot dont a besoin depuis le fichier dico.xml, le programme est en mesure de piocher aléatoirement un mot du niveau que le joueur aura indiqué.

Edition du dictionnaire

IV Cas test

Dans le cadre de nos test les noms de profil valable pour le moment sont profil et alban ce sont les seul qui ont un fichier xml contenant un profil qui existent pour le moment. Nous avons aussi teste les cas où on fournit un nom qui n'est associé a aucun profil et alors alors le programme renvoie au menu précèdent. Nous aussi décidé d'affecter des temps en fonctions des niveau de difficulté

Quelques images du jeu



Conclusion

Ce projet aura été l'occasion parfaite de maitre en œuvre et d'assimiler les notions XML ainsi que de APO vues en cours avec un cas pratique. Nous avons réussi a implémenter la plus part des fonctionnalité cependant il reste certains traitement comme l'affichage des score la fin du jeu ce qui reste un exercice intéressant que nous pourrons faire même si le projet s'arrête officiellement ici.