SBARTAÏ Mohamed Achraf
KADDOUR Mohammed

Rapport de projet Tux Letter Game

UE Formalisation Des Données - Technologies
XML L3 MIAGE

I. Introduction

Le projet que nous avions à réaliser cette année était un mini-jeu en 3D permettant d'appliquer toutes les notions de la programmation vues au cours de ce premier semestre.

Comme demandé, nous avons réalisé toutes les fonctionnalités de base du jeu.

Consignes de préparation :

Avant de pouvoir lancer le jeu, vous devez :

- récupérer le fichier src après avoir dezipper le fichier du rendu.
- Le fichier src contient le dossier game, XML, XSLT et test.
- Le dossier game contient toutes la classes java demandées.
- Le dossier XML contient tous les fichiers XML et elur schéma (XSD)
- Le dossier XSLT contient toutes les tranfomations xsl demandées.

Tux est un jeu dans lequel nous devons déplacer un personnage et récupérer des lettres pour pouvoir reformer un mot dans le temps imparti, en utilisant les commandes suivantes :

- Flèches de gauche et droite : déplacements horizontaux
- Flèches du haut et du bas : déplacements verticaux

Le mot que nous allons essayer de retrouver au cours d'une partie est choisi de façon aléatoire dans un dictionnaire, en fonction du niveau choisi par l'utilisateur.

Lors de l'identification de l'utilisateur, son profil est enregistré. Il pourra donc se reconnecter et jouer avec son profil. C'est pour cela que dans le menu principal, il peut :

- Créer un nouveau profil
- Charger un profil existant
- Ajouter des mots au dictionnaire
- Quitter le jeu

Nous avons décidé de classer les mots en 5 iférents niveaux :

- Niveau 1 :

  Le mot est entièrement donné avant le début de la partie, il est affiché pendant 5 secondes. Le joueur dispose de 30 secondes pour trouver le mot.

- Niveau 2 :

  Le mot est entièrement donné avant le début de la partie, il est affiché pendant 5 secondes. Le joueur dispose de 30 secondes pour trouver le mot.

- Niveau 3 :
  Le mot est entièrement donné avant le début de la partie, il est affiché pendant 5 secondes. Le joueur dispose de 30 secondes pour trouver le mot..

- Niveau 4 :
  Le mot est entièrement donné avant le début de la partie, il est affiché pendant 5 secondes. Le joueur dispose de 30 secondes pour trouver le mot.

- Niveau 5 :
  Le mot est entièrement donné avant le début de la partie, il est affiché pendant 5 secondes. Le joueur dispose de 30 secondes pour trouver le mot.

Une fois que l'utilisateur a créé ou chargé son profil, il peut commencer une nouvelle partie, en charger une ancienne, ou quitter le jeu. Dans le cas du chargement d'une partie, l'utilisateur verra ses dernières parties non terminées, et pourra choisir l'une d'entre elles et la rejouer.
La partie se termine lorsque le temps est écoulé, ou lorsque toutes les lettres ont été récupérées. Nous sommes ensuite redirigés sur le menu de jeu pour pouvoir refaire une partie, ou quitter.

II. XML et Parsings DOM/SAX

- Dictionnaire

Le dictionnaire est stocké dans un document xml, *dico.xml*. Afin de récupérer les mots qui y sont enregistrés dans notre application java, nous utilisons l'API SAX. Cette API utilise des événements gérés par l'application pour traiter un document xml. Le parseur SAX va parcourir le document xml dans son intégralité, et produira un événement à la rencontre d'un marqueur particulier dont nous aurons défini le traitement dans notre application.
A l'issue de ce parsing, tous les mots et leurs niveaux seront stockés sous forme d'objet dans la mémoire de notre application java, pour pouvoir être utilisés pour lancer les parties.

- Profil

Les profils des joueurs sont stockés dans des documents xml à leurs nom, par exemple *profil_Toto.xml*.
A la création de son profil par un joueur, son document xml est créé si le nom de joueur choisi est disponible (cad si aucun document xml ne porte déjà ce nom). Puis après chaque partie, ces dernières seront elles aussi enregistrées dans le profil xml du joueur.
Pour le chargement d'un profil, l'application va d'abord vérifier si un document xml ayant le nom demandé existe : s'il existe, le profil est chargé et le joueur est amené au menu de jeu, sinon le joueur est renvoyé au menu principal avec un message d'erreur.
Toutes ces opérations sont effectuées par un parser DOM : pour la création et la mise à jour du document xml du profil, les éléments sont créés et écrits dans le document xml par l'application java ; pour le chargement du profil, l'application lit les éléments du document xml par leur nom, et crée les

objets correspondant.

- Environnement

Le document *plateau.xml* contient toutes les informations nécessaires à la construction de l'environnement du jeu. Une classe Room implémente le parsing DOM de ces informations afin de récupérer les chemins des images de fond (textures) et les dimensions du plateau. Au lancement du jeu, le constructeur Room de cette classe est appelé, et par conséquent nous retrouvons les chemins et les dimensions dictées par le fichier xml dans notre jeu.

III. Problèmes rencontrés

a) Lorsque nous avons commencé à implémenter la partie Java de notre jeu et les collisions, nous nous sommes rendus compte qu'il y avait un problème majeur. En effet, si un mot comportait deux lettres identiques, pour réussir la reconstitution du mot, nous ne pouvions pas prendre n'importe laquelle des deux lettres. Cela était dû à l'association des lettres à un indice dans une liste de lettres du mot courant. Par exemple, prenons le mot "Technologie", le deuxième "o" étant associé à l'indice 8 il ne pouvait pas être récupéré par le Tux en première lettre du mot. Nous n'avons pas pu résoudre ce problème par manque de temps.

b) Concernant le parsing, nous avons rencontré plusieurs problèmes pour la sauvegarde des éléments, et l'ajout des différents éléments dans le fichier de profil. En effet, nous avons eu du mal à correctement ajouter les éléments, et après un appel de la fonction de sauvegarde le fichier était vide, sans parties sauvegardées ni même un profil. Nous avions simplement oublié de gérer nos erreurs dans le bloc catch de notre try catch, une fois que nous avons réussi à déceler l'erreur, notre fonction de sauvegarde était parfaitement fonctionnelle.

IV. Conclusion

Pour terminer, nous avons essayé à travers la réalisation de ce projet de comprendre le plus de principes de programmation liés au parsing, à la représentation et à l'organisation de données par un langage à balises, en l'occurrence XML.

Concernant le jeu, nous avons essayé de simuler une difficulté croissante à travers nos 5 niveaux, pour que chaque utilisateur ait un sentiment de progression au fur et à mesure des niveaux. Les fonctionnalités demandées pour le rendu final sont toutes implémentées, et le jeu fonctionne normalement.

Le jeu pourrait encore être amélioré, en ajoutant des changements de caméra pour Tux, des animations supplémentaires, une disparition de Tux pendant un moment lorsque l'on se trompe de lettre dans un mot. Ce jeu permet un grand nombre d'élargissements possibles !