Année scolaire : 2024-2025

Université Grenoble Alpes

Rapport Projet JeuVideo

Nom du jeu : Dragon's Wrath

Membres :

- **Moussa-Mohamed_Kalid**
- **MBAYAM_Hippoyte Djegomde**
- **Saidou-Momo_Abdoul-Kader**

Plan

I - Utilisation de XSD

Nous avons utilisé les fichiers XSD pour modéliser les différentes entités de notre jeu, telles que le joueur (*Player*), les projectiles (*Projectiles*), et les météores (*Meteors*). Les XSD utilisés nous ont permis de définir une structure précise et valide pour le fichier XML (myGame.xml), garantissant que les données respectent un format bien défini. À partir de ces schémas, nous avons généré des classes C# représentant ces entités, ce qui facilite leur manipulation sous forme d'objets dans notre programme. Les instances XML, créées selon ce modèle, décrivent l'état du jeu et peuvent être désérialisées en objets pour être utilisées directement dans l'application. Cette méthode assure une validation des données, améliore la lisibilité du code, et simplifie les processus de sérialisation et de désérialisation, rendant ainsi le développement plus robuste et organisé.

Fichier correspondant :

myGame.xsd
player.xsd
monster.xsd
meteor.xsd
rectangle.xsd
position.xsd
projectiles.xsd

II - Utilisation de XML & XSLT

L'intégration de notre fichier XML, myGame.xml, dans le projet de jeu vidéo est essentielle pour stocker et gérer les données du jeu. Ce fichier fournit les informations structurées nécessaires pour initialiser et maintenir l'état du jeu, comme la santé du joueur, les monstres, les météores, et les projectiles. Voici comment il est utilisé et exploité dans votre projet :
Le fichier myGame.xml est conçu pour représenter l'état initial du jeu et contient :

-Les données du joueur : Santé, attaque, position, taille.
-Les données des monstres : Santé, attaque, taille, position, vitesse.
-Les données des météores : Vitesse, position, état (IsOffScreen).
-Les données des projectiles : Taille, vitesse, position.
-Les informations globales : Score, timer de météores, état du jeu (menu, pause, etc.).

En résumé, myGame.xml joue un rôle central dans le stockage et la récupération des données du jeu. Avec l'intégration de myGame.xsl, vous obtenez également une visualisation claire et structurée de ces données. Cette approche combine le meilleur des deux : une gestion efficace des données XML et leur présentation conviviale via HTML.

La technologie XSLT a été utilisée pour transformer le fichier XML contenant l'état du jeu (myGame.xml) en un format HTML compréhensible et lisible par les utilisateurs. Le fichier XSLT correspondant, nommé myGame.xsl, se trouve dans le répertoire data/xslt pour assurer une organisation structurée des fichiers de transformation. Le fichier XSLT est appliqué à myGame.xml , et le résultat est un document HTML prêt à être affiché dans un navigateur. Nous avons utilisés cette technologie pour l'extraction ciblés des données. En effet, certaines informations spécifiques, comme la santé des monstres, la position des projectiles, ou encore le score du joueur, sont directement mises en avant. Le fichier myGame.xsl suit une approche structurée pour transformer les données XML. Voila ce que le fichier xslt affiche comme données.

Joueur :
Les informations sur le joueur (santé, attaque, vélocité, position) sont affichées dans une section dédiée. Les balises XML telles que mg:health, mg:attack et mg:position sont extraites et affichées.

Monstres :
Chaque monstre est traité avec un xsl:for-each, permettant d'afficher les informations de manière itérative : santé, attaque, vélocité, position.

Météores et projectiles :
De manière similaire, les sections des météores et des projectiles sont créées en parcourant les données XML et en affichant les propriétés pertinentes (vitesse, position, taille).

Informations globales :
Les données globales du jeu, comme le score et le temps écoulé, sont directement extraites de leurs nœuds respectifs (mg:score et mg:second).

Ainsi ,Le résultat HTML est bien structuré, ce qui facilite la compréhension des données du jeu.

Fichier correspondant :

myGame.xslt
myGame.xml
myGame.html

III - Utilisation de DOM & XmlReader

Dans le cadre de notre projet, nous avons utilisé les technologies DOM et XmlReader pour manipuler notre document XML(myGame.xml). Ces outils nous ont permis d'interagir efficacement avec la structure du fichier XML, que ce soit pour extraire des informations ou pour y apporter des modifications.

1. Extraction d'informations avec DOM

Nous avons utilisé DOM pour analyser (parser) le document XML et accéder à ses différentes données. Par exemple, nous avons extrait des informations clés comme :

- **Le nombre de projectiles :** en parcourant les nœuds correspondants au chemin my:`projectiles/my:projectile`, nous avons compté combien de projectiles étaient définis dans le fichier XML.
- **Le nombre de monstres :** un chemin similaire, my:`monsters/my:monster`, nous a permis de déterminer combien de monstres étaient présents.
- **Le nombre de météores :** en suivant le chemin my:`meteors/my:meteor`, nous avons pu compter les météores spécifiés dans le fichier.

(Dans la classe MyGameDomReader.cs)

Cette extraction a été réalisée de manière précise en utilisant les chemins XPath pour naviguer dans l'arborescence XML, garantissant une lecture rapide et fiable des données.

2. Ajout de données avec DOM

En plus de lire des informations, DOM a été employé pour ajouter dynamiquement des éléments dans le fichier XML à partir de notre programme en C#. Par exemple, nous avons pu :

- Ajouter de nouveaux monstres en spécifiant leur position dans l'espace.
  (Dans la classe MyGameDomReader.cs)

Le DOM nous a permis de manipuler directement les nœuds XML en ajoutant des éléments avec leurs attributs et leurs sous-éléments, ce qui a facilité l'intégration de nouveaux monsters dans le jeu.

3. XmlReader pour des performances optimisées

Pour compléter l'utilisation de DOM, nous avons employé XmlReader lorsque nous avions besoin de lire le fichier XML de manière séquentielle et performante. XmlReader nous a permis de :

- Parcourir le fichier XML rapidement sans charger l'intégralité du document en mémoire.
- Extraire des informations spécifiques tout en réduisant l'utilisation des ressources système.

IV - Sérialisation

Dans le cadre de notre projet, nous avons utilisé la sérialisation pour gérer efficacement les données de notre fichier XML. La sérialisation nous a permis de convertir notre fichier XML structuré en des objets de notre programme C#, ce qui facilite le chargement, le stockage et l'échange de données. Parallèlement, nous avons utilisé la désérialisation pour transformer les données du fichier XML en objets exploitables.

Cette méthode simplifie la manipulation de structures complexes, telles que les projectiles, les météores et les monstres. Par exemple, lors de la désérialisation, les attributs et éléments du fichier XML, comme la position, la santé ou la taille des entités, ont été automatiquement mappés à leurs équivalents dans nos classes C#. De même, la sérialisation a permis de sauvegarder les modifications apportées aux objets directement dans un fichier XML, en respectant les règles définies par le schéma XSD.

Grâce à cette approche, nous avons bénéficié d'une interaction fluide entre le programme et les données XML. Cela a non seulement assuré la persistance des données, mais aussi leur conformité avec la structure XML attendue, tout en réduisant le risque d'erreurs lors des opérations de lecture et d'écriture.

(Dans la classe MyGame_serealizer.cs)