Rapport de Projet : Shoot Them Up

Groupe :
ALLOUCHE Nesrine
IDJAKIRENE Anis
MOUMOU Nassima
BENAMMAR Mahmoud

Introduction :

Le projet **Shoot Them Up** est un jeu vidéo développé à l'aide de la bibliothèque **MonoGame** en **C#.** Le joueur contrôle un vaisseau spatial et doit survivre à une invasion ennemie en tirant des projectiles tout en évitant les collisions. Ce projet s'inspire du concept des jeux de tir "Shoot 'em up", tel que décrit sur la page Wikipédia dédiée, et a été conçu pour intégrer toutes les notions étudiées au cours, notamment **C#**, **XML**, **XSLT**, et d'autres outils de programmation.

Architecture et Structure du Code:

Le projet est structuré en plusieurs classes et fichiers représentant les différentes entités et mécanismes du jeu. Voici un résumé des composants principaux :

1. Classe Game1

La classe principale **Game1** gère le cycle de vie du jeu, y compris l'initialisation, la mise à jour et le rendu. Elle encapsule :

- Le joueur (Ship) : gestion de la position, des contrôles et des animations.
- Les ennemis (Enemies) : génération aléatoire et descente progressive.
- Les projectiles (Projectile) : gestion du tir et des déplacements.
- La gestion des scores : affichage du score et enregistrement des meilleurs résultats via XML.
- La logique de jeu : détection des collisions, gestion des vies et redémarrage après un Game Over.

---

2. Classe Ship

Cette classe représente le vaisseau contrôlé par le joueur.
Fonctionnalités :

- Mouvement basé sur les entrées clavier.
- Animation fluide à l'aide d'une spritesheet découpée en une grille de 3x3 frames.

---

3. Classe Enemies

Cette classe représente les ennemis qui apparaissent à intervalles réguliers.
Fonctionnalités :

- Déplacement vertical constant, avec une vitesse globale ajustable.
- Gestion des positions initiales générées aléatoirement.

---

4. Classe Projectile

Cette classe représente les projectiles tirés par le joueur.
Fonctionnalités :

- Mouvement rectiligne vers le haut.
- Normalisation des directions pour un déplacement constant.

Gestion des Ressources

1. Textures

Le jeu utilise plusieurs textures chargées depuis le répertoire Content :

- **Vaisseau** (ship) : spritesheet animée en 3x3 pour des transitions fluides.

- **Projectiles** (projectile1) : image statique représentant les tirs du joueur.

- **Ennemis** (enemy3) : image statique pour les vaisseaux ennemis.

- **Arrière-plan** (background) : texture utilisée comme décor principal.

2. Police de Caractère

La police DefaultFont est utilisée pour afficher les informations clés à l'écran, comme le score, les vies et les messages de **Game Over**.

Mécanismes de Jeu

1. Lors du lancement

Le jeu invite le joueur à entrer son nom. Cette fonctionnalité permet d’enregistrer le score avec le nom du joueur pour des futures références.

2. Système de Tir

Le joueur peut tirer des projectiles à l'aide de la touche **Espace**. Un système de délai (*cooldown*) est intégré pour limiter la cadence de tir à un projectile toutes les 0.3 secondes.

3. Apparition des Ennemis

Les ennemis apparaissent à des positions aléatoires en haut de l'écran à intervalles réguliers.

Difficulté dynamique :

- La vitesse des ennemis (Global Speed) augmente tous les 100 points de score.
- L'intervalle entre les apparitions diminue pour rendre le jeu plus intense.

4. Détection des Collisions

Deux types de collisions sont gérés :

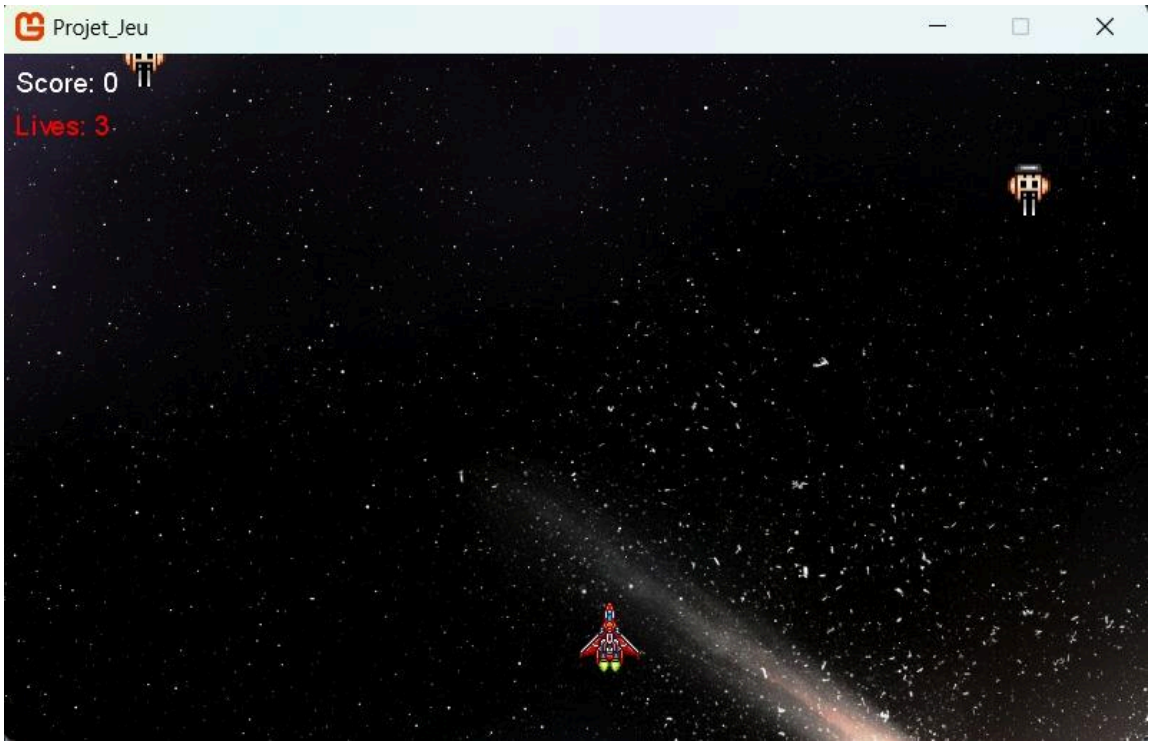
- **Projectile contre Ennemi** : supprime les deux entités et ajoute 10 points au score.
- **Ennemi contre Joueur** : retire une vie au joueur.

5. Écran de Game Over

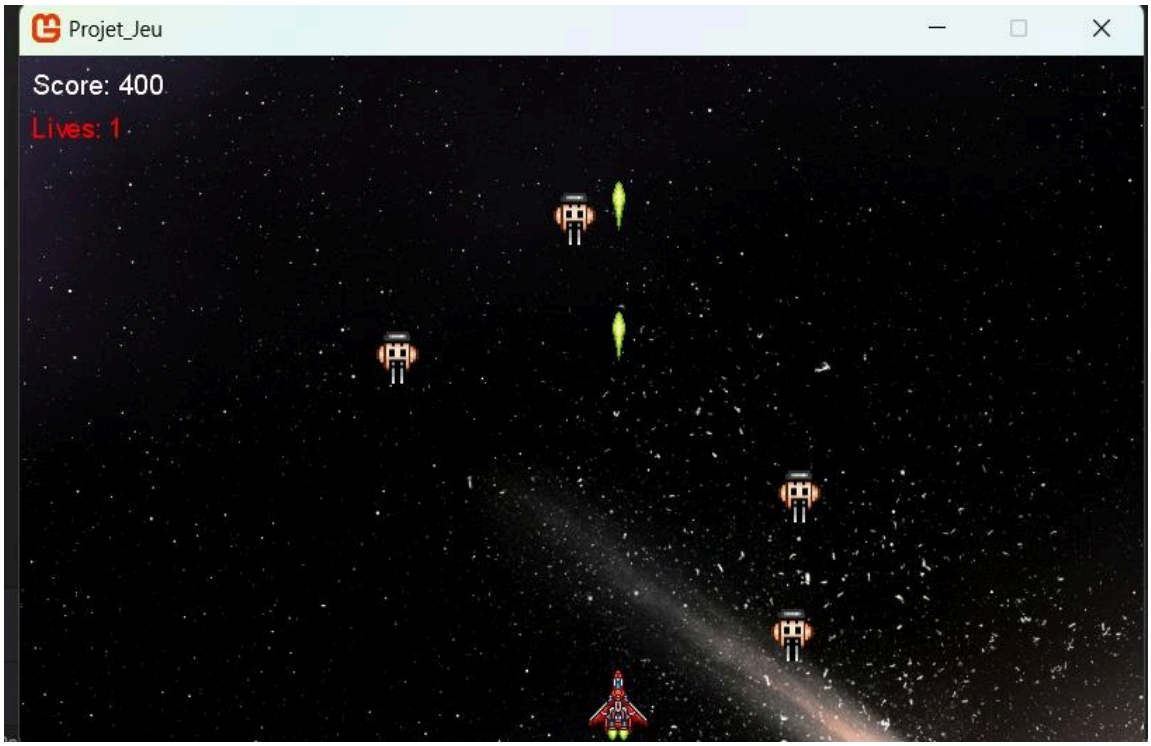
Lorsque le joueur perd toutes ses vies, l'écran affiche un message de **Game Over** et le score final. Le jeu peut être redémarré en appuyant sur la touche **R**.

6. interfaces du Jeu

- **Début du jeu**
  Voici l'interface affichée au lancement du jeu :



- **En cours de jeu**
  Voici l'interface lorsque le jeu est en cours :



Sérialisation:

1. Sérialisation en XML des Scores

La sérialisation permet de sauvegarder l'état des scores (pseudo, points, date, temps de jeu) dans un fichier XML. Ce fichier est ensuite relu lorsque le jeu redémarre, afin de conserver l'historique des performances des joueurs.

- Nous utilisons la classe **ScoreManager** qui encapsule la logique de sauvegarde et de chargement des scores. Elle s'appuie elle-même sur une classe générique XmlManager<T> pour lire/écrire les données sous forme d'objets C# sérialisés en XML.
- Le fichier Scores.xml contient la liste des scores (pseudo, score, date, temps), tous regroupés sous la racine **<Scores>**

La classe ScoreManager ajoute un score en appelant AddScore, puis écrit le fichier XML à l'aide de **XmlManager<T>.Save()**. À chaque partie terminée, les nouveaux scores sont ajoutés au fichier.

2. Validation XML par Schéma XSD:

L'intégrité du fichier XML est assurée par un schéma XSD (fichier Scores.xsd) qui définit les contraintes attendues :

- L'élément **<Score>** doit être un entier positif.
- La date doit être sous format YYYY-MM-DD.
- Le temps doit être un entier positif en secondes.

Lorsque le jeu charge les scores, il utilise **XmlManager<T>.Load()**. Avant de désérialiser, celui-ci valide le fichier XML contre le schéma XSD. Si le fichier ne correspond pas aux règles (par exemple, une date au mauvais format ou un score négatif), une exception est levée et le fichier n'est pas chargé, préservant ainsi l'intégrité des données.

3. Transformation XSLT

Afin d'avoir une vue plus conviviale des scores, une feuille XSLT (**Scores.xsl**) est utilisée pour transformer le fichier XML des scores en une page HTML. Cette page affiche un classement, le meilleur score, le top 5 et le joueur ayant passé le plus de temps en jeu(on s'est permis de faire une petite blague)

- La transformation XSLT est lancée après la sauvegarde, ce qui génère un fichier HTML comme par exemple ici: **Scores.html**
- On utilise des **<xsl:for-each>** et **<xsl:sort>** pour trier les scores. Le meilleur joueur, le temps le plus long et le top 5 sont ainsi mis en avant.

En résultat, le HTML produit permet de visualiser rapidement les scores des 5 meilleurs joueurs sur un navigateur web.

🏆 Le meilleur score: Nesrine

Top 5 des gens

pos	pseudo	score	date	temps
1	Nesrine	540	2024-12-15	67
2	anis	510	2024-12-15	60
3	Jean	320	2024-12-20	42
4	Nassima	300	2024-12-15	0
5	Alice	300	2024-12-20	60

Joueur le plus long à finir (ou pas): ***Nesrine***
(Franchement, chapeau, il a dû s'ennuyer)

4. Intégration au Jeu

La logique de gestion des scores est intégrée directement au code du jeu (classe Game1). Lorsque le joueur perd la partie :

- Le jeu calcule le temps écoulé à chaque partie jouée.
- Ajoute le score du joueur avec **ScoreManager.AddScore(...).**
- Le fichier XML est mis à jour et validé.
- La transformation XSLT peut être lancée automatiquement

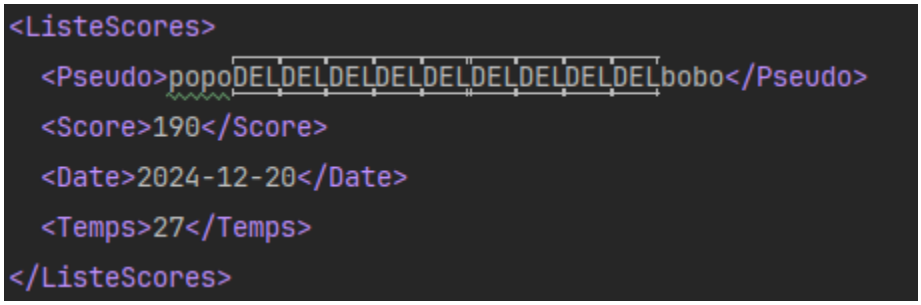
Cette approche sépare clairement les responsabilités : le jeu manipule les scores à travers **ScoreManager**, tandis que **XmlManager<T>** et le XSD assurent la validité technique et structurelle des données, et le XSLT fournit une vue lisible.

Problèmes rencontrés:

Durant la réalisation du jeu vidéo on a rencontré quelque problème qui sont:

1) dès le lancement du jeu on a pas pu afficher lorsque l'utilisateur entrait son pseudos dans le terminal et dès qu'il taper ok il fallait qu'il rentre une autre touche pour que le jeu se lance
2) Si l'utilisateur s'est trompé dans la saisie de son pseudo et il voulait supprimer quelque chose avec del il enregistre del dans les fichier Scores.xml exemple
3) Au vu du manque de temps nous n'avons pas pu rajouter un son pour le jeu

4) Pour lancer le jeu on a eu un probleme pour debloquer le fichier dotnet-tools.json exemple pour le débloquer : “Unblock-File -Path "C:\Users\Public\Documents\lalalo\ProjetJeu.config\dotnet-tools.json"



Conclusion :

Ce projet nous a permis de mettre en pratique les concepts appris, tels que la programmation orientée objet en C# et l’utilisation de technologies XML. Grâce à cette expérience, nous avons approfondi notre compréhension de la gestion de données, de la validation avec XSD, et de la transformation avec XSLT, tout en développant un jeu engageant et fonctionnel.

Ressources :

- Polycopié Cours :Formalisation des Données - Technologies XML (UGA, L3 Miage)
- https://fr.wikipedia.org/wiki/Shoot_%27em_up