Rapport technique
du projet jeu vidéo
Trivial Pursuit Remaster

rédigé par :
Doriane RICHARD, Lilian OURAHA, Mossab SADIKI

Règles du jeu

Le Trivial Pursuit Remaster est inspiré d'un jeu de société populaire, le Trivial Pursuit. C'est un jeu où les joueurs répondent à des questions de culture générale afin de récolter des morceaux de camembert.

Voici les règles du Trivial Pursuit Remaster :

Objectif du Jeu

L'objectif est d'être le premier joueur à obtenir six camemberts de différentes couleurs.

Déroulement du Jeu

- Lancer du Dé :
  Le premier joueur inscrit commence en lançant le dé grâce à la touche espace.

- Déplacement :
  Il s'effectue automatiquement en fonction du jet de dé. De ce fait, le pion du joueur avance automatiquement du nombre de cases indiqué par le jet de dé.

- Couleur de la Case :
  La couleur de la case sur laquelle le pion s'arrête détermine la catégorie de la question.

- Question :
  Vous devrez répondre à une question de la catégorie sur laquelle vous êtes tombé, cependant, vous pourrez tourner les choses à votre avantage en choisissant une difficulté entre : facile, moyen et difficile.

Si la réponse est correcte, le joueur gagné un camembert et la partie continue. Cependant, si la réponse est incorrecte, le tour passe au joueur suivant directement.

Catégories de Questions

Les questions sont réparties en six catégories, chacune représentée par une couleur différente :

- Histoire (Bleu clair)
- Jeux vidéo (Violet)
- Musique (Bleu foncé)
- Nature (Vert)

- Sports (Rouge)
- Zythologie (Jaune)

Type de case spécial

Si un joueur arrive sur une case joker, un joker va lui être assigné au hasard. Il y a deux types de joker :

- Relance de question permettra au joueur de piocher une autre question s'il a dû mal à répondre à la question posée.

- 50/50 : s'il est activé, il enlèvera deux réponses fausses, afin de permettre au joueur de n'avoir plus qu'une chance sur deux de trouver la bonne réponse.

Fin de Partie

Une fois qu'un joueur a récupéré ses six camemberts, il a gagné la partie.

Technologies demandées

De nombreuses technologies orientées données ont été utilisées. Cependant, bien que développées, toutes les fonctionnalités orientées données (XML et autre) n'ont pas été utilisées au sein même du déroulement du jeu. Cependant elles peuvent être utilisées hors partie pour visualiser des données par exemple.

Le dossier /chap/ contient la plupart des fonctionnalités non implémentées.

XML

Les fichiers XML sont disposés dans le répertoire */xml/*.

Ils ont servi à écrire nos données de manière structurée et centralisée dans un même répertoire.

Le fichier *Categories.xml* a servi à lister l'ensemble des catégories du jeu (Histoire, Nature, …)

Le fichier */xml/Cartes.xml* contient un grand nombre de données à savoir les Cartes. Chaque carte a des attributs, une question et une liste de 4 réponses. Nous avons en tout 288 réponses (6 catégories, 12 questions par catégorie, 4 réponses par question) ce qui justifie le grand nombre de données dans le fichier

Nous avons structuré les données du jeu dans plusieurs fichiers xml, par exemple le fichier game.xml regroupe les informations globales comme les joueurs, les catégories et le plateau et il constitue la base pour initialiser une partie. Le fichier case.xml décrit les détails des cases comme leur type “VIDE, CHANCE…”, leur couleur et la position, ce fichier sert pour configurer les cases sur le plateau. Et pour le fichier questionDesCategories.xml regroupe toutes les questions du jeu, chaque question inclut son texte, son niveau de difficulté et une liste de réponses possibles. et pour le fichier trivialpursuit.xml est généré automatiquement après la sérialisation, il représente l’état d’une partie incluant le plateau, les joueurs et les cartes des catégories et ce fichier permet de conserver une partie avec toutes ses informations. De même le fichier partie.xml est généré automatiquement après la sérialisation, il contient les informations sur les cases, leurs positions, leurs types et les cartes utilisées, il permet de sauvegarder les détails d’une partie.

XSD

Les fichiers xsd sont disposés dans le répertoire /xsd/.
Ils ont été utilisés pour définir la structure de nos fichiers xml.

Le fichier *Cartes.xsd* définit la structure d’un ensemble de cartes. Un élément cartes contient un ensemble de cartes, chacune d’entre elle étant identifiée par sa clé d’unicité liée à son attribut Id.

Le fichier game.xsd valide les relations entre les éléments globaux du jeu comme les joueurs, les catégories et les cases.
Le fichier catégorie.xsd assure que chaque catégorie inclut des cartes valides contenant des questions, des réponses et la difficulté.
Le fichier case.xsd définit les règles pour les cases par exemple il vérifie que la case possède un type valide, une position et une couleur.

Nous avons utilisé des clés d’unicité dans les fichiers xsd pour nous rassurer que certains éléments ou attributs des fichiers xml ne sont pas dupliqués, par exemple chaque joueur, chaque case ou chaque question devait avoir un identifiant unique. Par exemple dans game.xsd nous avons ajouté une clé d’unicité pour les identifiants des joueurs pour garantir qu’aucun joueur dans le fichier xml n’a le même identifiant. Et aussi nous avons utilisé des clés d’existence pour vérifier que certains éléments existent. Par exemple dans categorie.xsd nous avons ajouté cette clé pour garantir que chaque carte de question fait bien référence à une catégorie existante, cela signifie qu’aucune carte ne peut être associée à une catégorie qui n'existe pas dans le fichier xml

XSLT

Les fichiers xslt sont disposés dans le répertoire /xslt/.
Nous avons utilisé des fichiers xslt pour transformer les données xml en html. Par exemple, lorsque nous voulons afficher toutes les questions d’une catégorie, nous utilisons un fichier comme “questionHistoire.xslt”. Ce fichier parcourt les données les données xml et génère un tableau html avec les questions, leur difficulté et les réponses possibles. Chaque catégorie dispose de son propre fichier xslt, ce qui nous permet de générer des pages spécifiques pour chaque thème (exemple : Nature, zythologie, Histoire…). cette méthode

nous offre une manière simple et efficace de rendre les données lisibles. D'autre part, on a aussi des fichiers xslt permettent de générer des pages html affichant les détails de chaque type de case. Chaque transformation cible un type particulier de case, comme les cases de type VIDE, CHALLENGE, CHANCE…

HTML

Nous avons généré des fichiers html pour chaque thème de catégorie comme Musique, zythologie… Chaque fichier présente un tableau qui contient les questions, leur niveau de difficulté, et les réponses possibles avec une indication de la bonne réponse. Aussi, nous avons également généré des fichiers html pour représenter les cases du plateau selon leur type: CHANCE, VIDE, CHALLENGE… Ces fichiers décrivent chaque case par ses caractéristiques: sa couleur, sa position et sa taille. Et pour améliorer la présentation, nous avons utilisé un peu des styles CSS.
Ces fichiers html permettent de visualiser lrs données du jeu de manière claire. Et grâce à ces fichiers, l'accès aux informations nécessaires à la gestion ou l'analyse du jeu est facilité.

XMLReader

Nous avons utiliser XMLReader pour traiter les fichiers xml ligne par ligne sans charger tout le fichier en mémoire, cela nous a permis de parcourir efficacement les grands fichiers et de trouver des informations spécifiques comme extraire uniquement les cases de types VIDE tout en ignorant les autres cases.

Sérialisation / Désérialisation

Tout d'abord, nous avons une classe générique dans */chap/XMLUtils.cs* qui est utilisée pour sérialiser et désérialiser n'importe quel objet en XML.
Elle contient deux fonctions ***Deserialization*** et ***Serialization*** qui servent respectivement à sérialiser un objet et désérialiser un document
La raison de l'avoir rendue générique est tout simplement pour la rendre réutilisable et ne pas avoir à faire de fonction de Serialization / Deserialization pour chaque objet / XML.

La désérialisation a été utilisée dans le projet dans la phase d'initialisation du jeu, c'est-à-dire dans la fonction *LoadContent* de la classe principale.
Tout d'abord pour **désérialiser** l'ensemble des **catégories** avec leurs informations. Elles sont ainsi stockées dans une classe **Categories** dont l'unique utilité est sa désérialisation (elle ne contient qu'une liste de catégorie). Les données sont récupérées depuis le fichier */xml/categories.xml.*

La désérialisation est enfin utilisée pour récupérer l'ensemble des cartes du jeu liant chaque carte à une catégorie par son identifiant.Les données sont récupérées depuis le fichier */xml/Cartes.xml.* L'objet Carte étant lié à un objet Catégorie, pour

chaque carte sérialisé, on appelle une fonction GetCategorieViaId sur chaque Case pour lui lier l'objet Catégorie associé à l'id catégorie qu'il a récupéré.
La sérialisation de cartes était très intéressante dans notre contexte en vue des 288 réponses que nous avions.

Certaines données que l'on aurait voulu ne pas avoir à écrire dans "en dure" dans le code mais plutôt désérialiser n'ont pas pu l'être. C'est le cas de la classe **Case.**
En effet, dans la phase d'initialisation nous créons et instancions les 34 cases que comportent le plateau de jeu plutôt que de simplement les désérialiser depuis le fichier xml "case.xml".
La raison est la complexité de la classe, **Case** classe hérite de la classe **Sprite** et donc des attributs de **Sprite**. D'une part l'héritage en XML est parfois un peu laborieux et rendre la classe **Sprite** sérialisable l'est tout autant .

Parsing avec XmlSerilizer

Pour gérer la logique du jeu, nous avons modélisé chaque élément dans des classes en c#, par exemple, nous avons une classe Case qui représente chaque case du plateau, cette classe contient des informations comme le type de la case, sa couleur et ses coordonnées via un objet Sprite. De même, nous avons créé une classe Categorie pour représenter un thème, et une classe Carte pour modéliser les questions et leurs réponses… Nous avons également utilisé une classe appelée XMLUtils qui nous permet de sérialiser et désérialiser les données, par exemple, lorsque nous créons une nouvelle partie, nous pouvons la convertir en fichier xml pour la sauvegarder. Ensuite, si nous voulons charger une partie existante, nous utilisons la désérialisation.

Dans notre programme principal Program.cs, nous avons intégré des exemples concrets pour montrer comment les différentes parties du projet fonctionnent ensemble, par exemple, nous avons créé des objets représentant des cases, des catégories et des cartes. Ensuite, nous les avons sérialiser en fichier xml pour les sauvegarder. Nous avons également démontré comment utiliser XPath pour extraire des informations spécifiques comme le nombre de joueurs, les questions d'une catégories donnée. Nous avons utilisé le XPath pour permettre une navigation et une extraction précises des données xml.

DOM

La méthode DOM permet de charger tout le fichier xml en mémoire sous forme d'un arbre, cela nous donne une vue complète de toutes ses parties. Dans ce projet nous avons utilisé DOM pour récupérer des informations comme les joueurs, les catégories ou les cases du plateau. Grâce à DOM, on a aussi effectué des recherches précises comme trouver toutes les questions d'un thème ou compter le nombre de joueurs.

Autre :
Une maquette a été réalisée, afin de poser nos idées :
https://design.penpot.app/#/view/7ad540b5-8190-815d-8005-53f21544e7cd?page-id=7a

d540b5-8190-815d-8005-53f21544e7ce&section=interactions&index=0&share-id=f0fccb2e-2cc3-80b6-8005-6e5962e199eb

Il suffit de suivre cliquez sur le bouton ‘lancer la partie’ et de suivre le pion rouge, ou bien de cliquer sur les textes dorés en bas à droite.