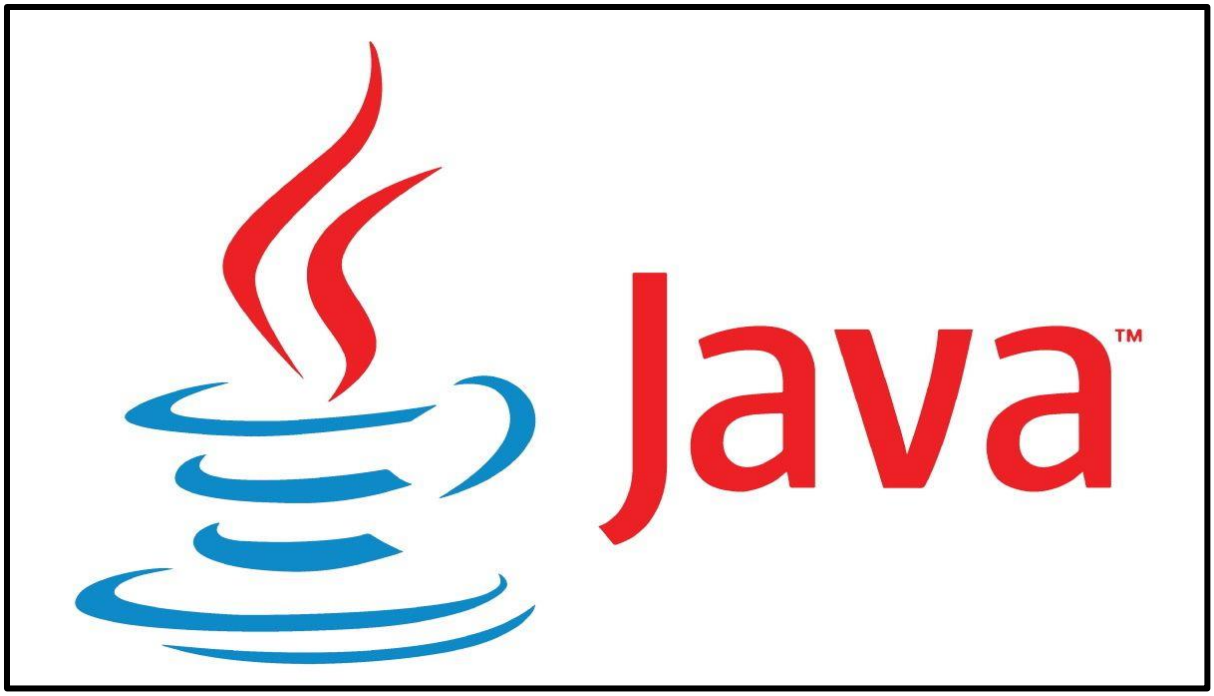
université
PARIS-SACLAY

Projet FeuFurieux



Equipe :

- CHABANE Oualid
- SAIL Ramy

Encadrée par : M. David REI.

2ème année licence informatique.

Année universitaire : 2023-2024

université
PARIS-SACLAY

Table de matières :

université
PARIS-SACLAY

1 Partie 01 : Introduction.

Notre projet Feufurieux est un jeu en deux dimensions, qui consiste à fuir à un feu qui s'élargie aléatoirement sur le terrain, et de trouver la porte de sortie, en passant par des obstacles comme des portes et de les ouvrir par des clés, aussi, notre jeu est doté d'une base de données des joueurs, leurs scores et leurs niveaux, et une possibilité de voir leurs classements selon les points obtenus.

2 Partie 02 : Analyse et conception.

2.1 Diagramme de classes :

Dans le pdf.

2.2 Découpage :

Le projet suit le modèle MVC, la raison pourquoi on à opté pour ce modèle est pour alléger le travail de groupe et d'éviter les conflits de Git, on a un modèle et un contrôleur afficheur.

Le modèle contient principalement les squelettes complétés et la classe de base de données (DataBase.java) qui nous permettre d'interagir avec les fichiers de base de données.

La base de données permet les joueurs avec leurs données et aussi les paramètres du jeu pour donner un aspect réalistique au jeu.

Le contrôleur afficheur n'est responsable que d'affichage, il contient les classes des trois pages de jeu qui sont, l'accueil, le menu des joueurs et la page principale du jeu (FenetreFeu étendu), ces trois classes échangent des données entre elles permettant pour le joueur de voir son classement, score et son niveau actualisé.

Le dossier assets permet de stocker toutes les icones, images et les audios utilisés dans le jeu.

3 Partie 03 : Réalisation.

3.1 Structure de la base de données :

La base de données contient deux fichiers binaire principaux, Joueurs.bin qui contient les joueurs et leurs données, ParametresJeu.bin qui contient pour l'instant que le nombre de joueurs totale, il peut être utile pour sauvegarder d'autres état de jeu dans le futur.

Les données des joueurs sont stockées par sérialisation, en d'autres termes, la classe Joueur implémente l'interface Serializable qui permet de stocker des objets comme des structures de données dans les fichiers

Pour optimiser les accès à la mémoire secondaire, on a utilisé des accès buffeurisés, en utilisant la clase BufferedInputStream et BufferedOutputStream de java, en plus, la sauvegarde des données se fait en fermant le frame du jeu dans l'autre part le chargement de données se fait en ouvrant le frame, par cette méthode on fait le nombre minimal d'accès mémoire.

3.2 L'interface graphique :

Les classes principales dans l'affichage sont celle d'accueil, menu et la page principale du jeu.

Pour rendre l'affichage plus agréable on affiche toutes les page du jeu dans le même frame donc chaque une des classes précédentes sauvegarde le frame dans ces attributs, une fois une classe est instanciée avec le frame du jeu passé dans ces paramètres du constructeur, l'écran se met à jour après l'ajout de l'élément graphique de la classe appelé à la liste ContentList du frame, bien évidement cela se fait après la suppression des éléments graphiques déjà existant dans la liste(les éléments de la page affiché en paravent).

La classe Accueil sert çà afficher l'écran d'accueil et passer le pseudo saisi par l'utilisateur à la classe Menu avec le frame du jeu crée dans son constructeur.

La classe Menu sert principalement à charger les données des joueurs à partir des fichiers vers la mémoire centrale, créer le joueur s'il n'existe pas, lancer le jeu et d'afficher les données de tous les joueurs dans un tableau ordonnées selon leurs niveaux et scores, et les données du joueur actuelle, pour mieux organiser le code, on a créé d'autres classes pour gérer cela :

La classe DataPane : sert à afficher les données actualisées du joueur actuelle et le bouton de lancement du jeu.

La classe EnsembleJoueurs : sert à initialiser, mettre à jour et afficher la base de données dans un tableau ordonnée selon le score des joueurs et leurs niveaux, l'implémentation de l'interface Comparable est primordiale pour ordonner les joueurs dans le TreeSet, si deux joueurs ont le même score et niveau, on les ordonne selon leurs identifiants.

La classe PageJeu sert à afficher le jeu avec des icones un peu réalistiques, la résistance et le nombre de clés du joueurs actualisées et le bouton de retour vers le menu, parmi les attributs le plus important le cette classe est le temporisateur, il se déclenche une fois la page monte, et s'arrête une fois la page descende, cela se fait en implémentant WindowListener.

Pour afficher le joueur toujours centré, on fait une translation par un vecteur

V= (Vx ; Vy) qui permet de déplacer le joueur et ses voisins à une case fixe dans la fenêtre d'affichage, voici une figure qui visualise la translation :

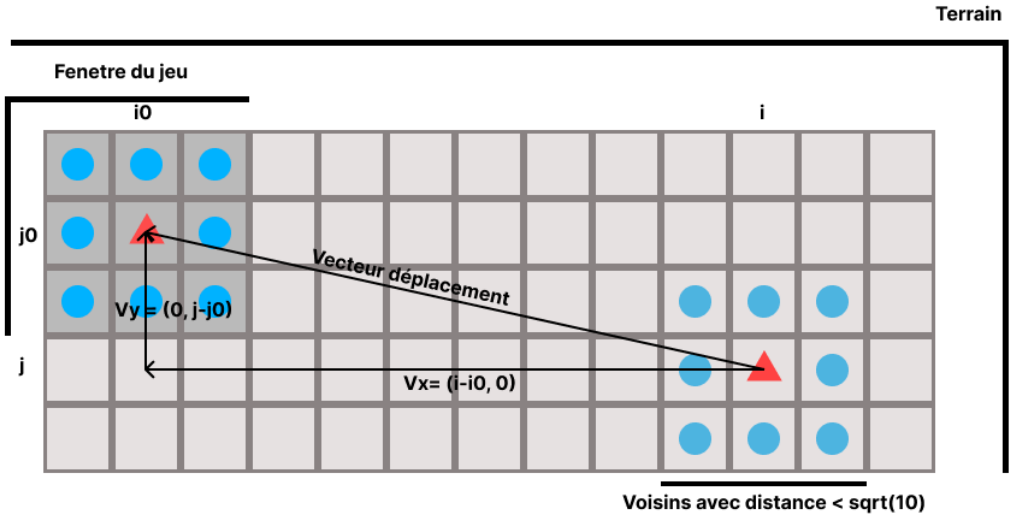


Figure : Visualisation de la manière d'affichage d'un joueur et ses voisins.

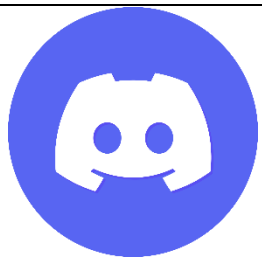
3.3 Le modèle :

Le modèle contient les classes qui manipulent les données des joueurs et du terrain.

Les portes sont considérées comme des cases traversables.

Pour ordonner les joueurs dans tableau de menu, on doit les identifier d'une manière unique, pour le faire, on a rajouté n attribut statique qui sert à compter le nombre des joueurs (ce numéro peut être utilisé comme numéro d'un nouveau joueur) et un fichier binaire pour stocker cet attribut pour éviter d'avoir un conflit en cas d'introduction d'un nouveau joueur après une la réouverture du jeu (ParametresJeu.bin).

4 Partie 04 : Conduite du projet.



Notre choix pour le modèle MVC nous à permet de travailler d'une manière libre, on à utilisé une chaine discorde pour les réunions, la communication, des suggestions, subdiviser les taches, …

En premier, on a subdivisé les taches d'une manière générale puis on a travaillé ensemble dans les premiers jours, après l'avancement dans le projet, les taches sont devenues plus claires ce qui a permet à chaque un de nous de travailler d'une manière asynchrone.

On a fait quatre réunions, la première c'était dans le premier jour pour discuter les fonctionnalités de bases du projet.

Dans la deuxième, on a subdivisé les taches définitivement et décidé d'ajouter les trois pages pour le jeu.

La troisième réunion était pour discuter sur les effets sonores et les images sur le terrain pour rendre le jeu plus agréable.

En fin, la dernière est programmée pour s'entrainer pour la présentation.

5 Partie 05 : Conclusion.

En conclusion, ce projet nous a permet d'apprendre pleins de principes de langage java et le paradigme orienté objet aussi de nous introduire à la programmation graphique, On a essayé d'utiliser le maximum de principes de ce langage, comme le polymorphisme d'objets et de méthodes (surcharge et redéfinition), l'abstraction surtout dans les classes d'interface graphique, l'encapsulation et bien évidement l'héritage.