RAPPORT SAÉ 3.1

Dorfromantik

IUT Sénart-Fontainebleau

David Akagunduz, Vincent Teissier, Bamba Top

Sommaire

Introduction

Dans ce rapport, nous vous exposons le développement d'un jeu basé sur Dorfromantik.

Pour créer un paysage harmonieux, le joueur est chargé d'assembler des tuiles hexagonales. Le jeu débute avec une première tuile placée automatiquement. À chaque tour, une tuile est dévoilée au joueur qui détermine la position et l'orientation de pose de celle-ci. L'unique restriction est que la nouvelle tuile doit se trouver à proximité d'une tuile existante.
La partie prend fin après la pose de 50 tuiles, et le score du joueur est déterminé en additionnant les points acquis pour chaque poche sur le terrain.

Fonctionnalités du programme

Menu principal

Le menu principal de l'application sert à choisir une série à effectuer pour commencer une nouvelle partie. Ce menu est conçu pour mettre le joueur dans l'ambiance de Dorfromantik, avec un arrière-plan qui nous rappelle un paysage. Le design permet au joueur de naviguer facilement avec un bouton pour jouer, un pour savoir comment jouer, un pour quitter.

Jouer

Le bouton Jouer permet de choisir une série parmi un nombre de séries et de lancer une partie.

Comment jouer ?

C’est un bouton qui sert de guide. En effet, une interface où l’on peut trouver des explications concernant le « gameplay » du jeu.

Quitter

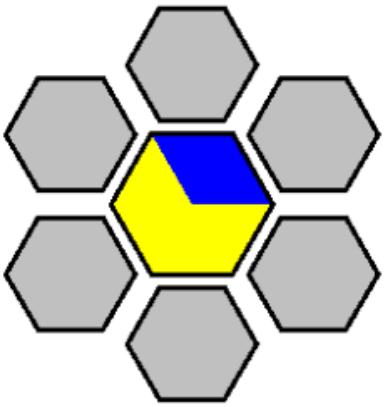
C’est un bouton qui sert à quitter le jeu depuis le menu. Il est utile dans les cas où le jeu a été lancé accidentellement.

Mouvement de la caméra

Pour que le joueur puisse se repérer facilement dans la fenêtre, il est possible de déplacer la caméra en utilisant la clique droite de la souris.

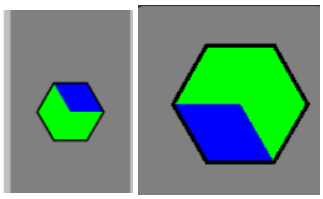
Poser une tuile

Le joueur peut appuyer sur des hexagones qui servent à poser une tuile.



Rotation d’une tuile

Pour que le joueur puisse améliorer sa gestion des tuiles, il est possible qu’il puisse tourner une tuile afin de pouvoir l’insérer quelque part.

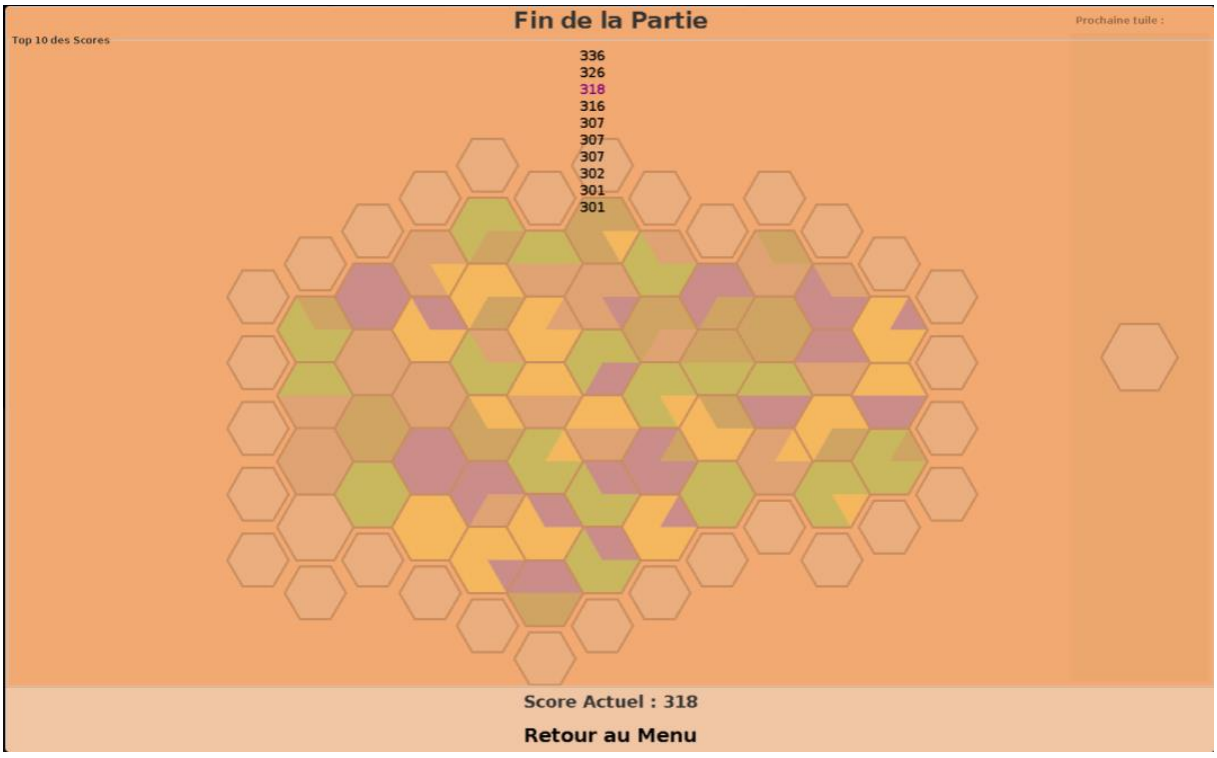


Comptage de score

Le score est affiché durant la partie entière, il augmente en fonction des tuiles posées.

Menu de fin

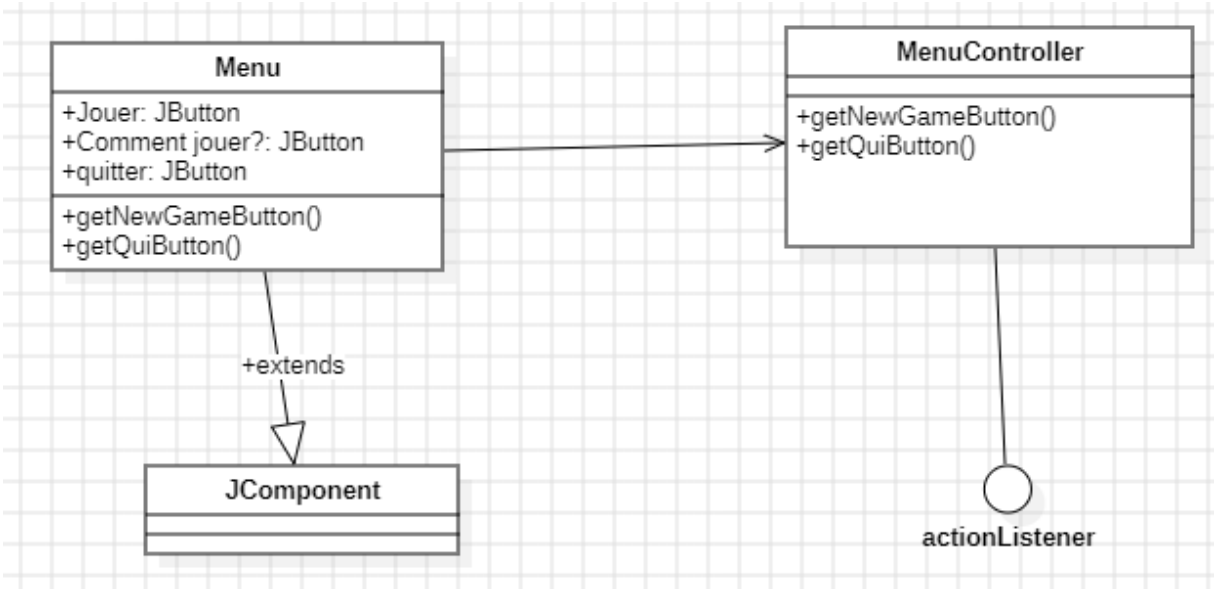
Une fois la partie terminé, un menu s'affiche, l'utilisateur voit son score final et peut choisir entre quitter où retourner dans le menu principal pour tenter une autre série.



Structure du programme

Menu principal

Le menu a été conçu d'une manière très simple. Des boutons qui à l'aide d'un contrôleur effectue des taches différentes.



JEU

Nous avons adopté une architecture MVC (Modèle-Vue-Contrôleur), ce qui nous a permis de structurer le code de manière organisée et modulaire. L'utilisation de cette architecture nous a permis de séparer les responsabilités entre la logique métier, l'interface utilisateur, et les contrôles, facilitant ainsi la collaboration en équipe et la lisibilité du code.

Le modèle représente les données de notre application et les règles de gestion du jeu. Il contient les classes qui définissent les éléments principaux du jeu :

Tile.java : Cette classe représente les tuiles du jeu, chaque tuile étant composée de différents types de terrain.

TerrainType.java : Enumération définissant les différents types de terrain possibles pour chaque tuile (forêt, prairie, mer, etc.).

Pocket.java : Cette classe gère les "pochettes" de terrains connectés, permettant de calculer des points en fonction des tuiles connectées entre elles.

TileDatabaseManager.java : Responsable de la gestion et de la manipulation des données des tuiles, cette classe est essentielle pour le chargement et la sauvegarde des informations des tuiles dans le jeu.

En rassemblant ces éléments dans le modèle, nous avons pu centraliser toute la logique métier du jeu et les données, ce qui facilite les modifications et les ajouts sans toucher aux autres parties du code.

Le dossier view contient toutes les classes qui gèrent l'interface utilisateur. L'objectif de ce dossier est de séparer l'affichage graphique et les interactions visuelles du reste du code. Les classes de ce dossier incluent :

GameView.java : La vue principale où le jeu est affiché et où les interactions de l'utilisateur se produisent.

MenuView.java et HowToPlayView.java : Ces classes gèrent respectivement l'affichage du menu principal et du panneau "Comment jouer".

HexagonTile.java : Gère l'affichage des tuiles hexagonales, en représentant les terrains par des segments colorés.

ScoreView.java : Affiche le score en temps réel, offrant une vue claire des points accumulés.

App.java : La fenêtre principale qui initie et organise toutes les vues.

Le dossier img contient les images nécessaires à l'interface, et les éléments personnalisés comme BtnPerso.java et ButtonHoverListener.java sont utilisés pour styliser les boutons et ajouter des effets visuels. Cette séparation permet d'améliorer la lisibilité et de simplifier la maintenance de l'interface.

Le dossier controller contient la logique de contrôle, reliant le modèle et la vue. Ce dossier inclut les classes et listeners nécessaires pour gérer les actions de l'utilisateur, comme les clics et les mouvements de souris :

GameController.java : La classe principale qui coordonne les actions du jeu et la communication entre le modèle et la vue. Elle gère la logique du jeu, comme la pose des tuiles et la vérification des règles.

GameContext.java et ScoreGameContext.java : Gèrent le contexte du jeu (la grille de jeu, le calcul des scores) et permettent de vérifier les connexions entre les tuiles.

CameraControllerListener.java et MouseWheelController.java : Contrôlent le déplacement et le zoom de la caméra, offrant à l'utilisateur une navigation fluide sur la grille.

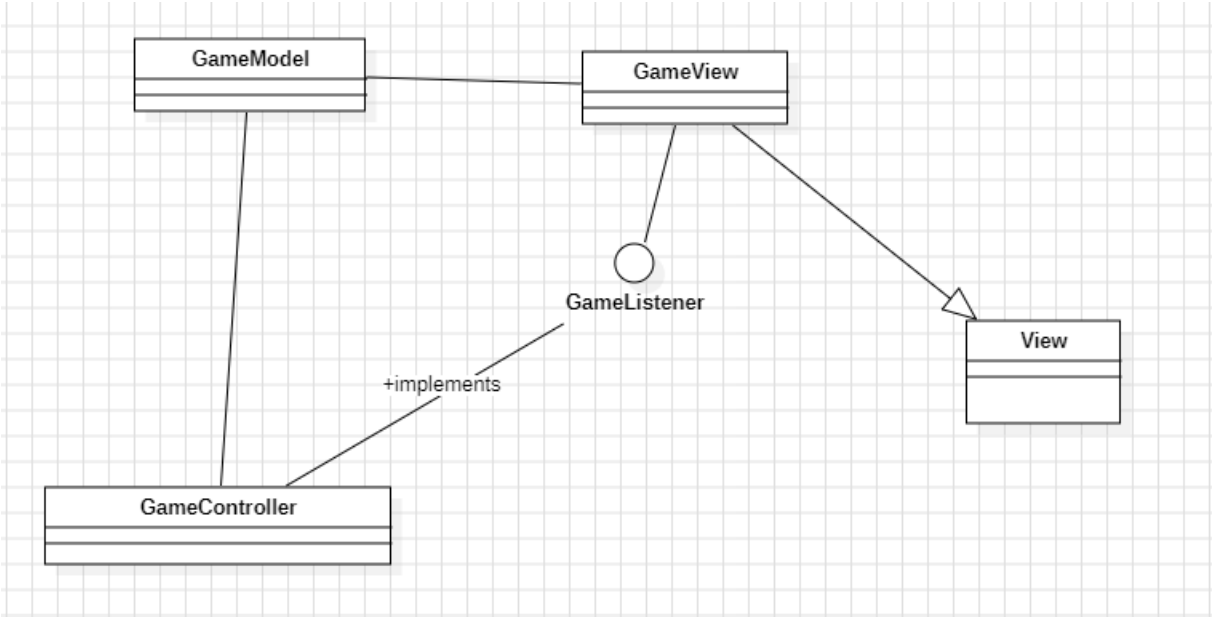
HexagonMouseListener.java, MouseDragHandler.java, MousePressHandler.java : Gèrent les interactions de la souris pour sélectionner des tuiles, les déplacer, et interagir avec la grille.

SerieListener.java et SeriesSelector.java : Gèrent la sélection des séries de jeu via le menu.

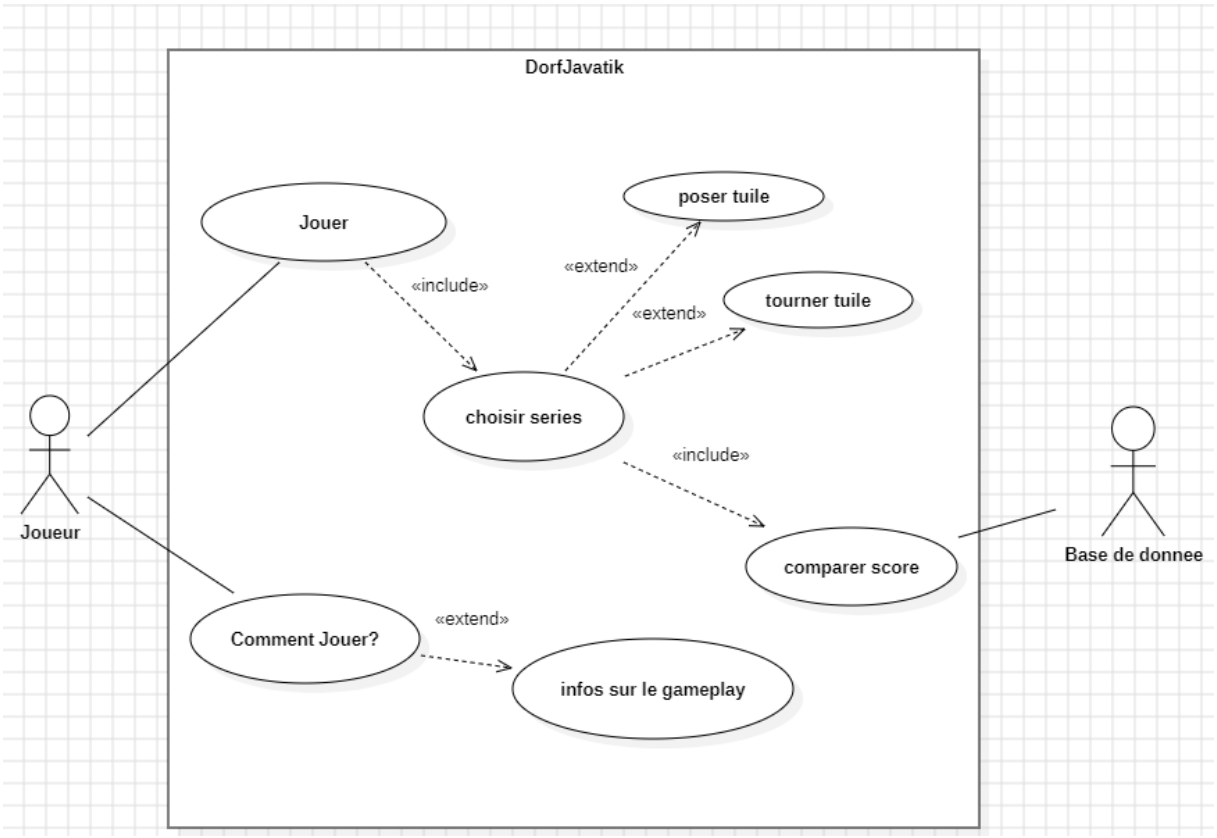
Les contrôleurs agissent comme intermédiaires, en coordonnant les interactions de l'utilisateur avec l'affichage graphique et en appliquant la logique métier nécessaire pour que le jeu réagisse correctement.

Dossier Music : Ce dossier contient les fichiers audios, permettant d'ajouter une ambiance sonore au jeu pour enrichir l'expérience utilisateur.

Voici le diagramme démontré de la manière la plus simple possible :

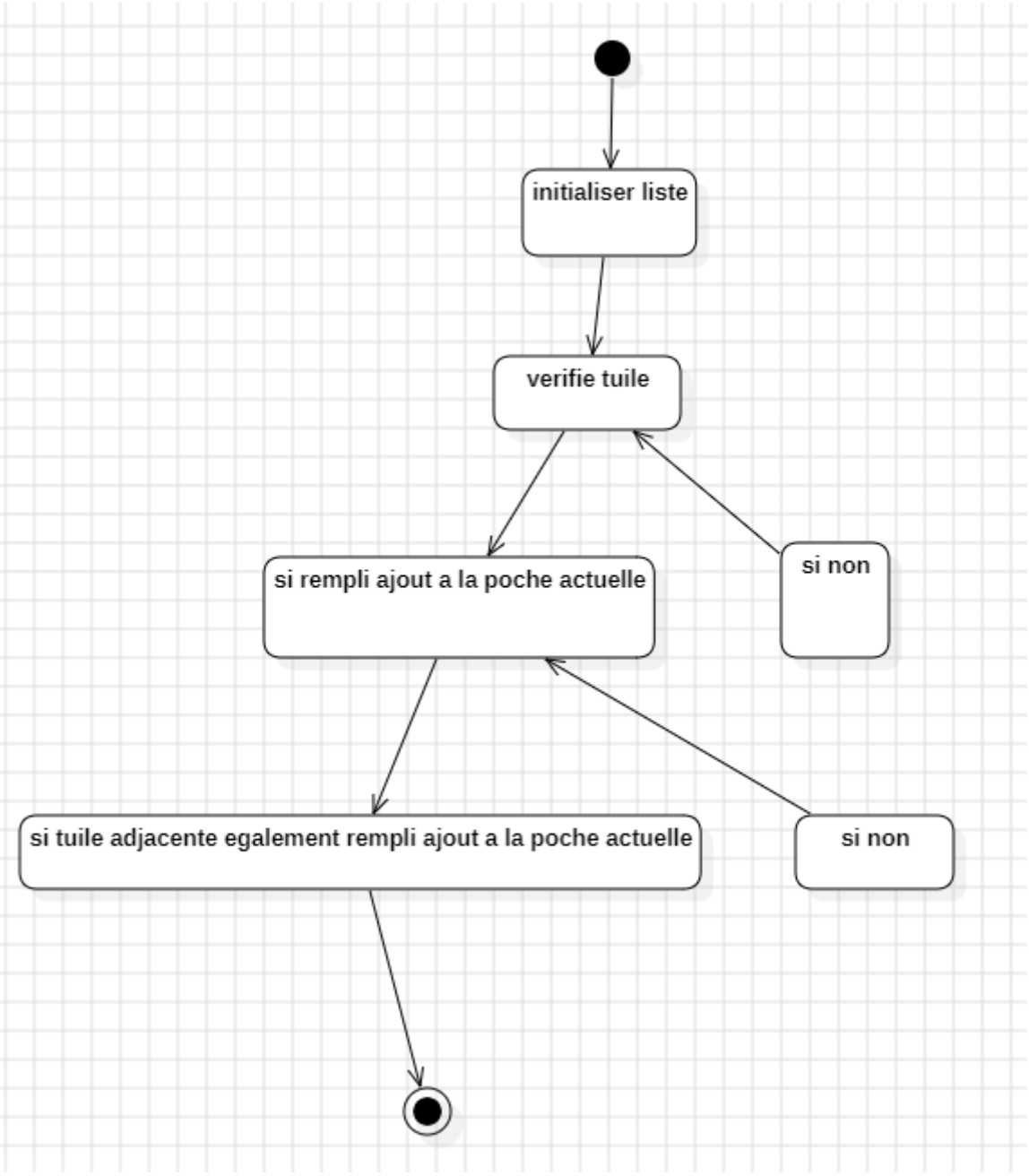


Un petit DCU pour expliquer notre point de vue :



Algorithme

L'algorithme pour identifier les poches dans le code HexagonTile peut se baser sur la méthode de recherche en profondeur (DFS) ou en largeur (BFS). Chaque tuile hexagonale est parcourue pour vérifier ses segments de terrain. Lorsqu'une tuile remplie est rencontrée, on explore ses tuiles adjacentes et on regroupe celles qui partagent des caractéristiques similaires (type de terrain). Ces groupes, une fois identifiés, représentent les "poches" sur le plateau, permettant ainsi de les mettre en évidence visuellement en utilisant la méthode setContrastColor.



Explication des écrans

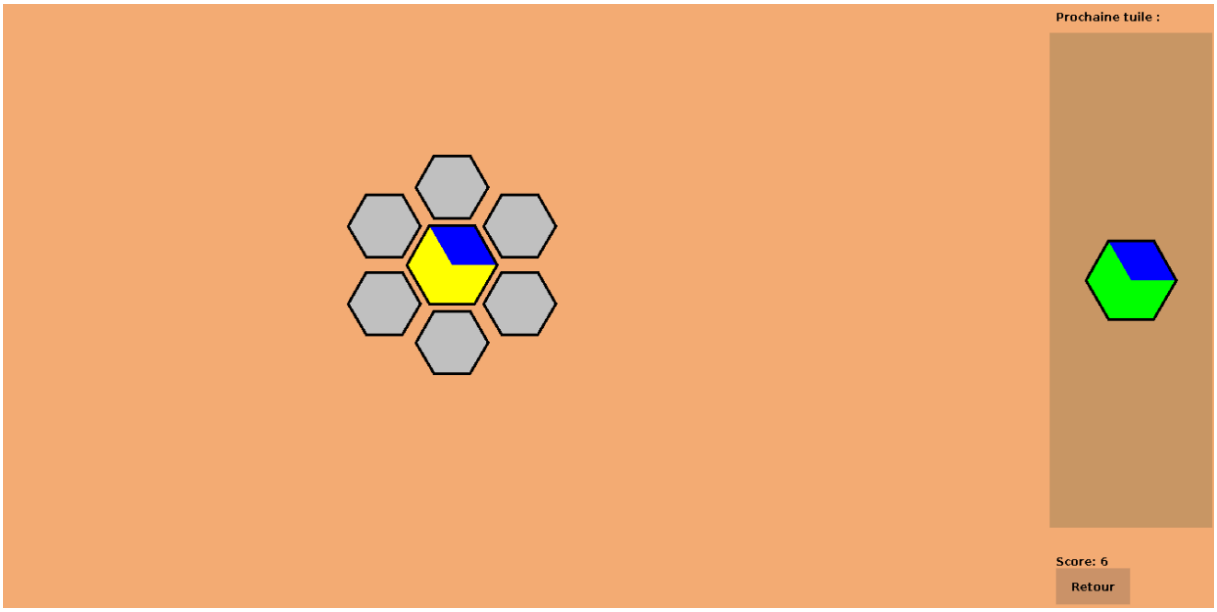
Nous avons fait le choix d’un design minimaliste et simple qui permet aussi une compréhension simple et rapide de l'utilisateur.

Menu principal

L'objectif du menu est d'immerger le joueur dans une aventure, avec un paysage en fond qui évoque la gestion d'un territoire, accompagné d'une musique de fond dans le même thème. Les teintes ont été sélectionnées de manière que les boutons soient clairement perceptibles. Pour la première fois, nous avons utilisé un "Paint Component" pour gérer l'affichage de ses boutons. Cependant, après avoir effectué des vérifications, nous avons constaté que la fenêtre manquait de responsivités. C'est ainsi que nous avons opté pour un 'GridBagLayout' pour gérer le menu. On a décidé de mettre les boutons " JOUER" et "COMMENT" à gauche car les personnes lisent en F et ainsi attirer leur vision à cet endroit.

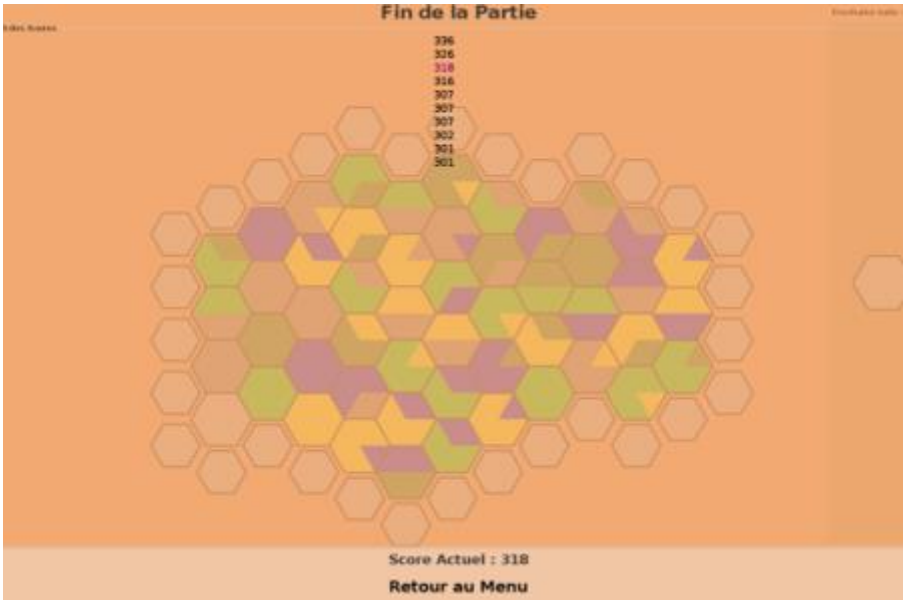
Jeu

En ce qui concerne l'interface du jeu en elle-même, la dimension des tuiles était pensée pour que le joueur ne soit pas perturbé visuellement, mais nous avons été obligés de les augmenter pour que le joueur ait l'impression de gérer un grand territoire. Par conséquent, les tuiles cliquables permettant de placer des tuiles collées aux véritables tuiles pouvaient causer une confusion au joueur. C'est pourquoi nous avons choisi de diminuer leurs tailles. Nous avons donné la possibilité à l'utilisateur de se déplacer dans le jeu et ainsi ne pas forcer l'utilisateur a posé au lieu qui lui est visible ce qui lui laisse sur liberté de jeu. D'ailleurs les lieux des tuiles qui sont possible sont plus petite et en grise afin de permettre à l'utilisateur de ne pas le confondre avec une tuile déjà posée. On a aussi en quelque sorte diviser la page avec donc le jeu et la prochaine tuile posée



Comparaison du score

A la fin d'une partie, on affiche le top 10 des scores si votre score fait partie des 10 il est mis en évidence en violet au contraire s'il n'est pas dans le top 10 ou même s'il l'est on affiche votre score. Cela vous permet de comparer votre score au meilleure et ainsi savoir quel est objectif que vous devez atteindre.



Conclusion

Akagunduz David :

J'ai beaucoup apprécié travailler sur le projet inspiré du jeu Dorfromantik. La phase de compréhension des fonctionnalités a été relativement simple grâce à la convivialité du jeu. De plus, ayant découvert Dorfromantik un mois avant le début de la SAE, je me sentais serein quant à la complexité du code, pensant que cela ne me prendrait pas trop de temps. Cependant, je me suis rendu compte que j'avais sous-estimé les défis à relever.

Au cours de ce projet, j'ai souvent été bloqué par des erreurs d'inattention, mais cela m'a également permis d'apprendre davantage sur le langage Java. Je ressens une certaine frustration de ne pas avoir réussi à bien développer l'algorithme de calcul du score. Malgré cela, cette expérience a été enrichissante et m'a permis d'améliorer mes compétences en travail d'équipe. Pour notre prochain projet, nous veillerons à commencer plus tôt afin d'optimiser notre temps et d'éviter les imprévus.

Teissier Vincent :

Ce projet de jeu inspiré de DorfRomantik a vraiment été une aventure. Je suis fier du résultat, surtout parce qu'on a tout généré en code, sans images, et ça me tient vraiment à cœur. Le projet a été un vrai défi, mais on a appris à organiser notre code proprement avec des packages et des classes séparées ; avant, on avait tendance à tout mélanger. Travailler dessus a été à la fois difficile et fun, et finalement, c'est un jeu vraiment addictif. Le travail en équipe n'a pas toujours été simple, mais on a codé sans relâche et partagé des idées en pair programming. Je suis content de ce qu'on a accompli ensemble, et ce projet est un résultat dont on peut tous être fiers.

Top Bamba :

Ce projet m'a vraiment aidé à mieux comprendre le développement sur JAVA, surtout en ce qui concerne le Modèle-Vue-Contrôleur (MVC). Même si ce projet n'était pas aussi amusant que d'autres sur lesquels j'ai travaillé, il m'a donné l'opportunité d'apprendre plein de nouvelles techniques de programmation. J'ai compris l'importance de structurer le code de façon plus claire, en séparant bien la logique de l'application de l'interface utilisateur, ce qui rend le tout beaucoup plus facile à gérer et à faire évoluer.

Travailler en équipe a également été enrichissant pour moi. J'ai réalisé à quel point il est important de partager ses idées, d'écouter les retours des autres et de respecter les délais fixés ensemble. Cette expérience m'a préparé à mieux gérer de futurs projets en groupe, en me donnant une compréhension plus claire des dynamiques d'équipe et des défis techniques. Je me sens désormais plus à l'aise avec mes compétences.

RAPPORT SAÉ 3.1