

Projet java

Treasure Hunt



Mohamed Dhia Eddine Thabet
2 info D

| | |
|--|----------|
| Introduction | 2 |
| Outils Utilisé | 4 |
| JavaFX | 4 |
| IntelliJ IDEA | 4 |
| Chaîne YouTube "RyiSnow" | 4 |
| ChatGPT | 4 |
| Diagramme de class | 5 |
| Travail Demandé Selon le Cahier des Charges | 5 |
| l'héritage restreint | 5 |
| Record | 5 |
| Diviser le mini projet en au moins deux packages et un module | 6 |
| Implémenter au moins un try with ressources pour vos exceptions | 6 |
| interfaces fonctionnelles | 6 |
| expressions Lambda ou références aux méthodes | 6 |
| Collections | 7 |
| Les streams et les expressions Lambda et/ou références aux méthodes pour les traitements sur les collections | 7 |
| Une interface graphique codée en JavaFx | 7 |
| TableView | 7 |
| Guide d'utilisation du jeu "Treasure Hunt" | 7 |
| 1. Ouvrir le jeu | 8 |
| 2. Cliquez sur "Start" | 8 |
| 3. Chercher les clés et les bonus | 8 |
| 4. Ouvrir les portes | 8 |
| 5. Trouver le trésor | 8 |
| 6. Recommencer le jeu pour un meilleur score | 8 |
| 7. Voir votre tableau des scores | 8 |
| Conclusion | 9 |

Introduction

"Treasure Hunt" est un jeu d'aventure conçu pour immerger le joueur dans une quête palpitante où exploration, stratégie et résolution de problèmes sont essentielles. L'objectif principal du jeu est de collecter un certain nombre de clés dispersées à travers une carte labyrinthique. Ces clés permettent de déverrouiller des portails, ou "gates", qui bloquent le chemin vers la sortie.

Le jeu est structuré autour de plusieurs niveaux ou missions, chacun présentant des défis uniques tels que des obstacles à contourner, des ennemis ou des pièges à éviter, et des ressources limitées à gérer. Le joueur doit donc faire preuve d'intelligence pour trouver le chemin optimal tout en surmontant les difficultés.

La victoire est atteinte lorsque le joueur réussit à collecter toutes les clés nécessaires, déverrouille les portails, et s'échappe de la carte avant la fin du temps imparti (si applicable). Cette dynamique ajoute une tension constante et motive les joueurs à améliorer leurs performances.

Au cours du développement du jeu "Treasure Hunt", j'ai utilisé **des tutoriels disponibles sur YouTube** pour améliorer divers aspects du gameplay et de la mécanique du jeu. Ces ressources m'ont été particulièrement utiles pour comprendre et implémenter la détection des **collisions**, qui s'est avérée être l'une des étapes les plus complexes du projet. Grâce à ces vidéos, j'ai pu apprendre des techniques avancées pour gérer les interactions entre les objets du jeu, garantissant ainsi une expérience fluide et réaliste pour le joueur.

"Treasure Hunt" offre ainsi une expérience interactive et immersive, combinant réflexion, exploration et amusement. Le jeu a été conçu en Java avec une attention particulière portée à l'optimisation des performances et à l'ergonomie de l'interface utilisateur, garantissant une prise en main intuitive et une fluidité de gameplay.

L'**histoire de ce jeu est inspiré par l'anime**

The promised Neverland



Outils Utilisé

Pour concevoir et développer le jeu *Treasure Hunt*, les outils et ressources suivants ont été utilisés :



JavaFX a été utilisé pour développer l'interface utilisateur graphique (GUI), permettant la création d'éléments interactifs et une expérience utilisateur immersive adaptée aux besoins du jeu.



Cet environnement de développement intégré (IDE) a facilité l'écriture, l'organisation, et le débogage du code, tout en optimisant le processus de développement.

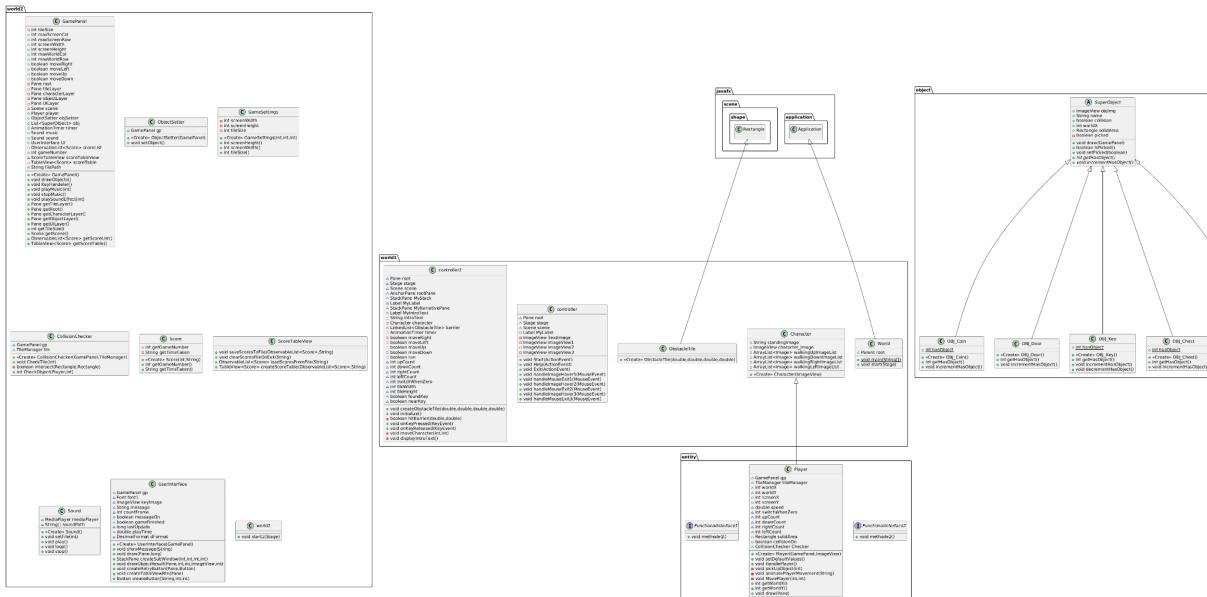


Les tutoriels de cette chaîne, bien qu'originellement basés sur Swing, ont servi de guide pour comprendre des concepts essentiels tels que la gestion des événements et la détection des collisions.



J'ai utilisé ChatGPT pour traduire et adapter les concepts et le code basés sur Swing des tutoriels de RyiSnow en JavaFX, ce qui a simplifié la transition vers un cadre plus moderne et mieux adapté à mon projet.

Diagramme de class



Travail Demandé Selon le Cahier des Charges

l'héritage restreint

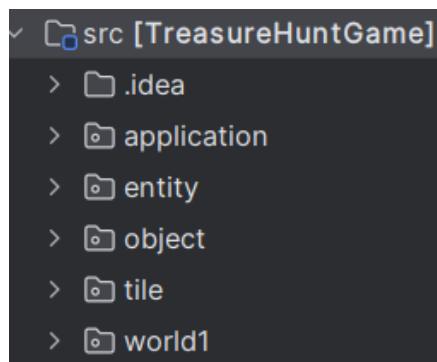
Class **Character**

```
public sealed class Character permits Player
```

Record

```
public record Score(int getGameNumber, String getTimeTaken) ... 10 usages
```

Diviser le mini projet en au moins deux packages et un module



Implémenter au moins un try with ressources pour vos exceptions

Class TileManager

```
public void loadMap() { 1 usage
    try {
...
} catch(IOException e) {
    e.printStackTrace();
}
```

interfaces fonctionnelles

```
public interface Functional { 2 usages 1 implementation
    public void methode(); 1 usage 1 implementation
}
```

expressions Lambda ou références aux méthodes

```
Functional w = () -> System.out.println("Starting menu");
```

```
Functional w = new Functional() {
    @Override 1 usage
    public void methode() { System.out.println("Starting menu");}
};
```

Collections

```
public ArrayList<Image> walkingUpImageList; 6 usages
public ArrayList<Image> walkingDownImageList; 6 usages
public ArrayList<Image> walkingRightImageList; 6 usages
public ArrayList<Image> walkingLeftImageList; 6 usages
```

Les streams et les expressions Lambda et/ou références aux méthodes pour les traitements sur les collections

```
Map<Class<?>, List<SuperObject>> groupedObjects = gp.obj.stream().filter(obj -> obj.name!="Door" )
    .collect(Collectors.groupingBy(SuperObject::getClass, Collectors.toList()));

List<Map.Entry<Class<?>, List<SuperObject>>> list = new ArrayList<>(groupedObjects.entrySet());

list.sort((entry1, entry2) ->
    entry2.getValue().size() - entry1.getValue().size()
);
```

Une interface graphique codée en JavaFx

TableView

```
public class ScoreTableView { 2 usages
    public TableView<Score> createScoreTable(ObservableList<Score> scoreList) { 1 usage
        TableView<Score> tableView = new TableView<>();

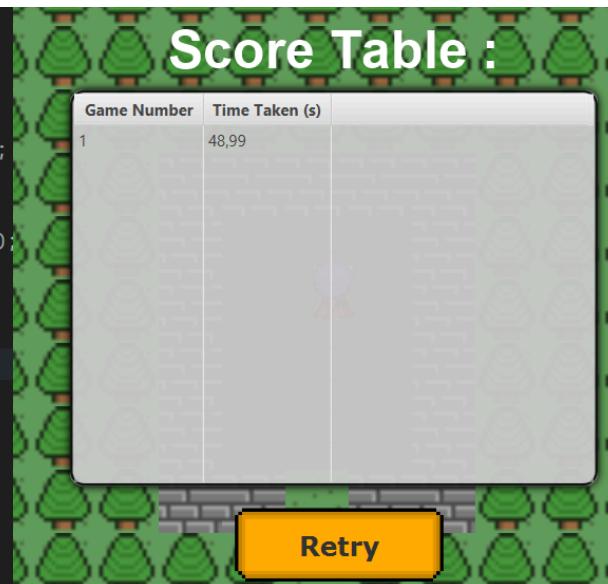
        TableColumn<Score, Integer> gameNumberColumn = new TableColumn<>( s: "Game Number");
        gameNumberColumn.setCellValueFactory(new PropertyValueFactory<>( s: "GameNumber"));

        TableColumn<Score, String> timeTakenColumn = new TableColumn<>( s: "Time Taken (s)");
        timeTakenColumn.setCellValueFactory(new PropertyValueFactory<>( s: "TimeTaken"));

        tableView.getColumns().clear();
        tableView.getColumns().add(gameNumberColumn);
        tableView.getColumns().add(timeTakenColumn);

        tableView.setItems(scoreList);

        return tableView;
    }
}
```



Guide d'utilisation du jeu "Treasure Hunt"

1. Ouvrir le jeu

Lancez le jeu *Treasure Hunt* sur votre appareil. Attendez que le menu principal apparaisse, où vous pourrez commencer votre aventure.

2. Cliquez sur "Start"

Dans le menu principal, cliquez sur le bouton "**Start**" pour commencer votre quête. Une fois cliqué, le jeu se chargera et vous serez transporté sur la carte.

3. Chercher les clés et les bonus



Votre tâche principale est de chercher les clés disséminées sur la carte. Déplacez-vous et cherchez-les dans des endroits cachés ou difficiles d'accès. Collectez-en autant que possible !

4. Ouvrir les portes



Après avoir collecté les clés, cherchez les portes verrouillées qui nécessitent ces clés pour s'ouvrir. Utilisez les clés pour déverrouiller les portes et explorer de nouvelles zones de la carte.

5. Trouver le trésor



L'objectif ultime est de trouver le trésor caché quelque part sur la carte. Une fois que vous avez déverrouillé suffisamment de portes et exploré les zones, dirigez-vous vers l'emplacement du trésor pour le récupérer et terminer la partie.

6. Recommencer le jeu pour un meilleur score

Retry

Après avoir trouvé le trésor, vous pouvez recommencer le jeu pour essayer de battre votre score précédent. Cherchez plus efficacement, collectez plus de clés et ouvrez les portes plus rapidement pour obtenir un meilleur score.

7. Voir votre tableau des scores

Scores

Après chaque partie, consultez le tableau des scores pour voir vos résultats. Votre score est basé sur votre rapidité à trouver les clés, à ouvrir les portes et à atteindre le trésor.



| Score Table : | |
|---------------|----------------|
| Game Number | Time Taken (s) |
| 1 | 307,06 |

Conclusion

En conclusion, *Treasure Hunt* est un jeu captivant qui met à l'épreuve la rapidité et la stratégie des joueurs. À travers la collecte de clés et l'ouverture de portes pour découvrir le trésor, le jeu propose une expérience interactive et stimulante. Les fonctionnalités de la carte générée aléatoirement, associées à la recherche des clés et à la gestion du temps, permettent de garantir une rejouabilité élevée. Grâce à l'intégration de **JavaFX** pour l'interface et l'utilisation de diverses ressources, notamment des **tutoriels** et l'aide de **ChatGPT**, le développement du jeu a permis d'approfondir la compréhension de la programmation et de la gestion des événements dans un environnement de jeu.

Ce projet a été une expérience enrichissante, tant sur le plan technique que créatif, et a permis de mieux apprêhender les défis liés à la conception d'un jeu vidéo simple mais **amusant**. J'espère que les joueurs apprécieront l'expérience et prendront plaisir à améliorer leurs scores tout en explorant ce monde virtuel.

De plus, **je prévois d'ajouter de nouvelles missions** pour enrichir l'expérience de jeu et offrir davantage de contenu aux joueurs. Mon objectif est de **publier** *Treasure Hunt* sur **Steam** afin de toucher un plus large public et d'élargir son potentiel. J'ai également des projets pour ajouter une dimension **narrative** au jeu, en créant une histoire créative inspirée par l'anime **The Promised Neverland**, qui s'alignerait avec la philosophie du jeu, apportant ainsi une profondeur émotionnelle et narrative à l'expérience. Cette évolution du jeu serait une étape majeure dans son développement et un moyen de rendre l'aventure encore plus captivante.

