L3 INFORMATIQUE

Arthur Haye

Rapport de projet

Debut Avril 2023 a 30 Mai 2023

Tuteur du projet : *Mr X. Skapin*

Etablissement : *Universite de Poitiers – Prometee*

SOMMAIRE

Realisation du projet :

Pour réaliser ce projet, j'ai décidé de réaliser tout le code en java, cela me paraissait le plus simple pour concilier la partie oriente objet l'IHM. J'ai réfléchi à quoi ressemblerait ce projet et la conception de celui-ci, a la partie orientée objet et l'IHM ainsi qu'à la base de données. J'ai réalisé les différents schéma et modèles permettant de réaliser ce projet, cela a permis de développer ce projet au mieux. J'ai réalisé ce projet en essayant de me rapprocher le plus possible d'un jeu Diablo dans la mesure du possible. Ce que j'ai réalisé a été conforme à mes attentes, j'ai développé la conception du jeu pour qu'il ait beaucoup d'interface et de fonctionnalités, ainsi qu'une base solide pour développer celui-ci. Vous trouverez ci-dessous ce que j'ai réalisés pour la conception du jeu, cela comprend plusieurs schémas et idées qui m'ont permis d'atteindre cet objectif. Cela comprend un diagramme de classes assez complexe que j'ai réussi à améliorer au fil du temps, comprenant de nombreuses classes. La partie IHM de ce rapport sera compose de captures d'écrans de l'interface et du code.

Presentation du projet :

Dans notre projet, on incarne Lucifer, celui-ci peut se trouve sur une carte de 10x10 cases, le héros peut se déplacer d'une case a une autre dans 8 directions différentes grâce à des boutons de déplacements.

Sur la carte se trouve le décor, cela peut être simplement le sol sur lequel le heros marche mais egalement des obstacles. Il y a egalement des items qui peuvent etre trouves au sol qui peuvent etre ramasses par le heros et qui seront places dans un inventaire qui sera ouvrable par le joueur avec un bouton, l'inventaire affichera la liste des items detenus par le heros et les actions que l'on peut faire avec (l'utiliser ou le jeter).

Les unites autre que le heros sont les ennemis, ceux-ci viendront attaquer le heros et lui faire baisser sa vie, si le heros veut vaincre les ennemis, il doit se mettre a portee d'eux ou utiliser des competences qui coûteront du mana au heros, ces competences seront sous forme de boutons.

Il y a plusieurs types d'ennemis avec chacun des statistiques differentes. Le heros possede un systeme de niveau, s'il tue des ennemis, sa barre d'experience montera si elle atteint le montant maximum, elle se reset et le heros gagne un niveau. La barre d'experience et le niveau du heros sont affiches a cote de la carte, certaines competences du heros se debloqueront en fonction du niveau de celui-ci, il gagnera egalement des statistiques en montant de niveau.

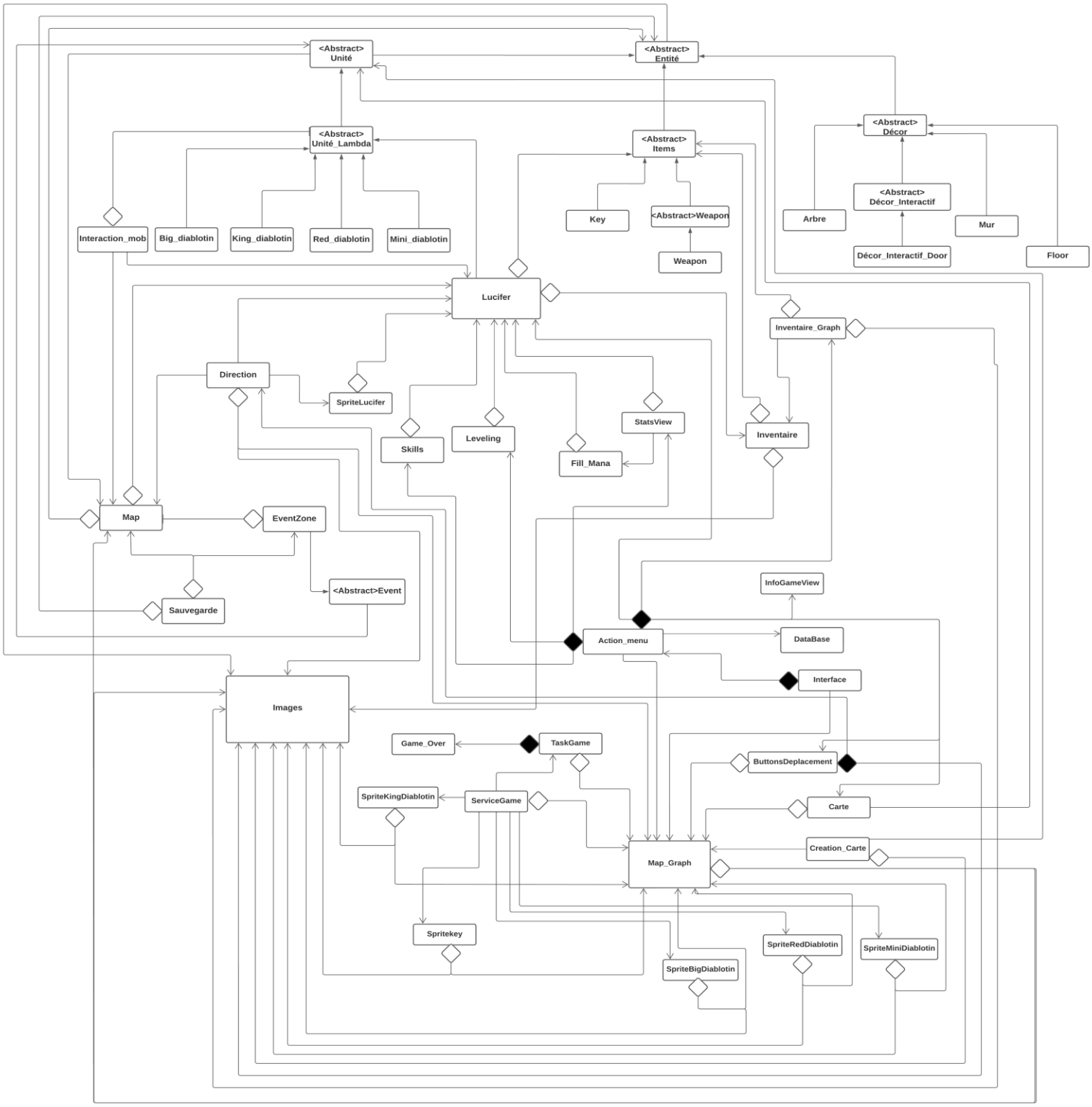
Les statistiques du heros seront affiches dans l'interface, le heros a des points de vie, de mana, une portee, une vitesse et des degats.

Le mana du heros se remplit progressivement avec le temps. Il y a egalement un menu dans l'interface qui repertorie toutes les actions, par exemple si le heros ramasse une cle, un message apparaîtra dans ce menu, il permet egalement de suivre les actions pendant un combat, quand un ennemi attaque le heros et inversement, des messages apparaîtront et indiqueront les degats infliges et les PV restant des unites qui combattent. Le joueur peut ouvrir une carte pour avoir une vue plus en profondeur sur ce qui entoure le heros, il peut l'ouvrir grace a un bouton sur l'interface.

Il y a aussi un bouton qui permet au joueur de sauvegarder sa partie pour pouvoir la reprendre plus tard. Chacune des images du jeu sont dans une classe Images qui possede toutes les images du jeu, chacune des classes necessitant ces images pourront y acceder et ainsi constituer l'aspect graphique du jeu.

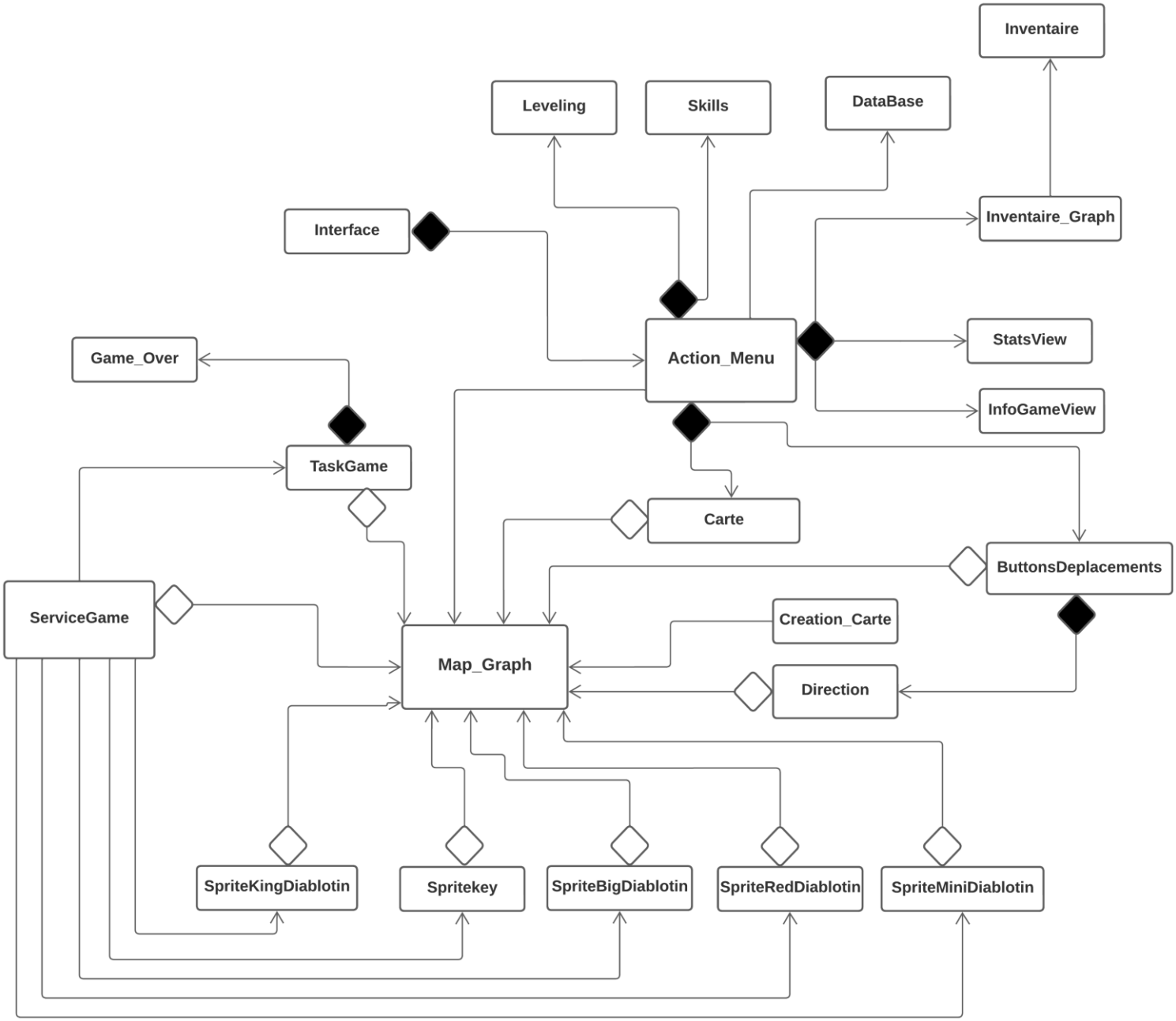
Diagramme de classes :

Diagramme du projet complet :



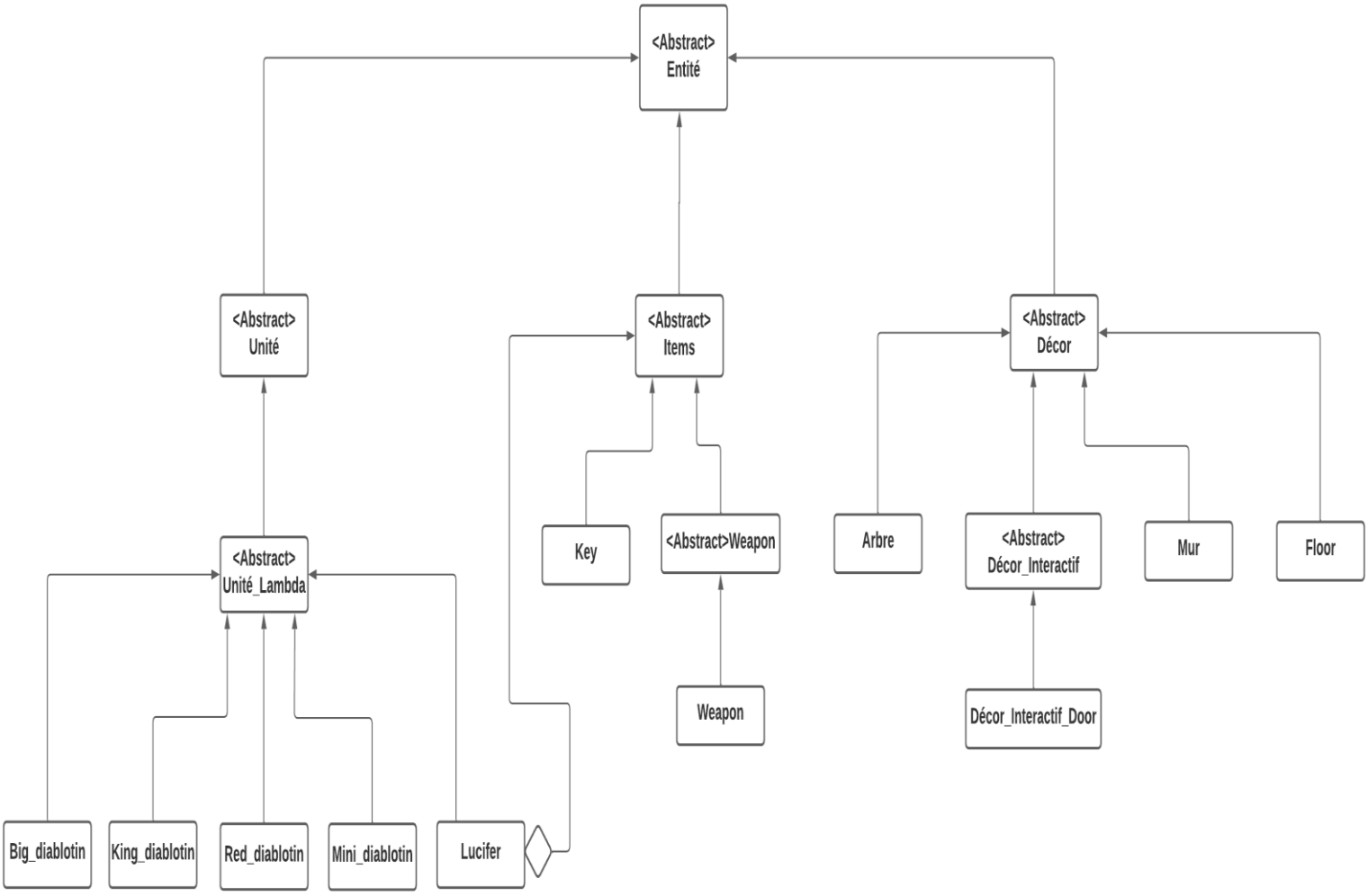
Schemas simplifies :

(Il n'y a pas toutes les classes mais les meilleurs pour une comprehension globale) Principales classes de l'interface du jeu : Map_graph correspond a la carte du jeu, Action_Menu represente l'interface en jeu avec tous ses elements comme les menus de stats, les skills, la carte, etc. Interface utilise Action_Menu qui sera ensuite utilise dans le Main pour lancer le projet.



Hierarchie de la classe entite :

La classe entite represente toutes les entites qui seront presentes dans le jeu, avec les unites, que ce soit le heros ou les ennemis, les items qui seront sur la map ou dans l'inventaire du heros et les decors qui composeront la map, dont ceux avec lesquels on pourra interagir comme les portes.



Interface homme-machine :

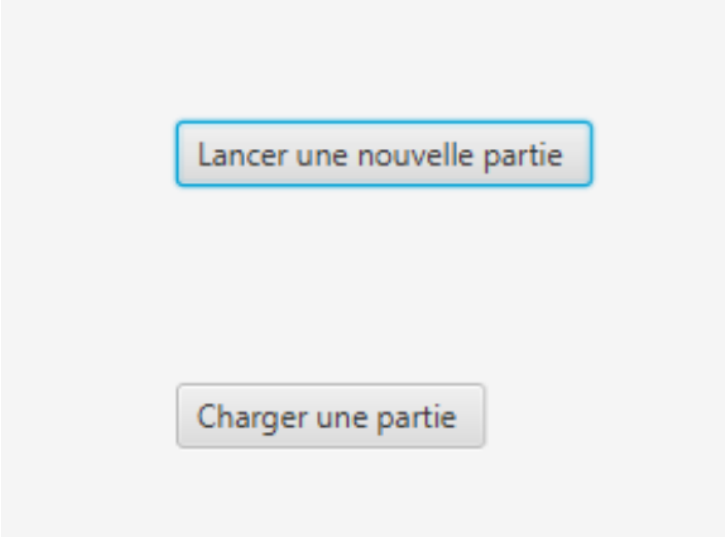
Boutons permettant soit de lancer une nouvelle partie soit d'en charger une que l'on a prealablement sauvegardee :

Lancer une nouvelle partie : Charge ici une nouvelle partie a partir d'une map preconcu via notre Outils de Creation_Map, l'utilisateur commence son experience ici.

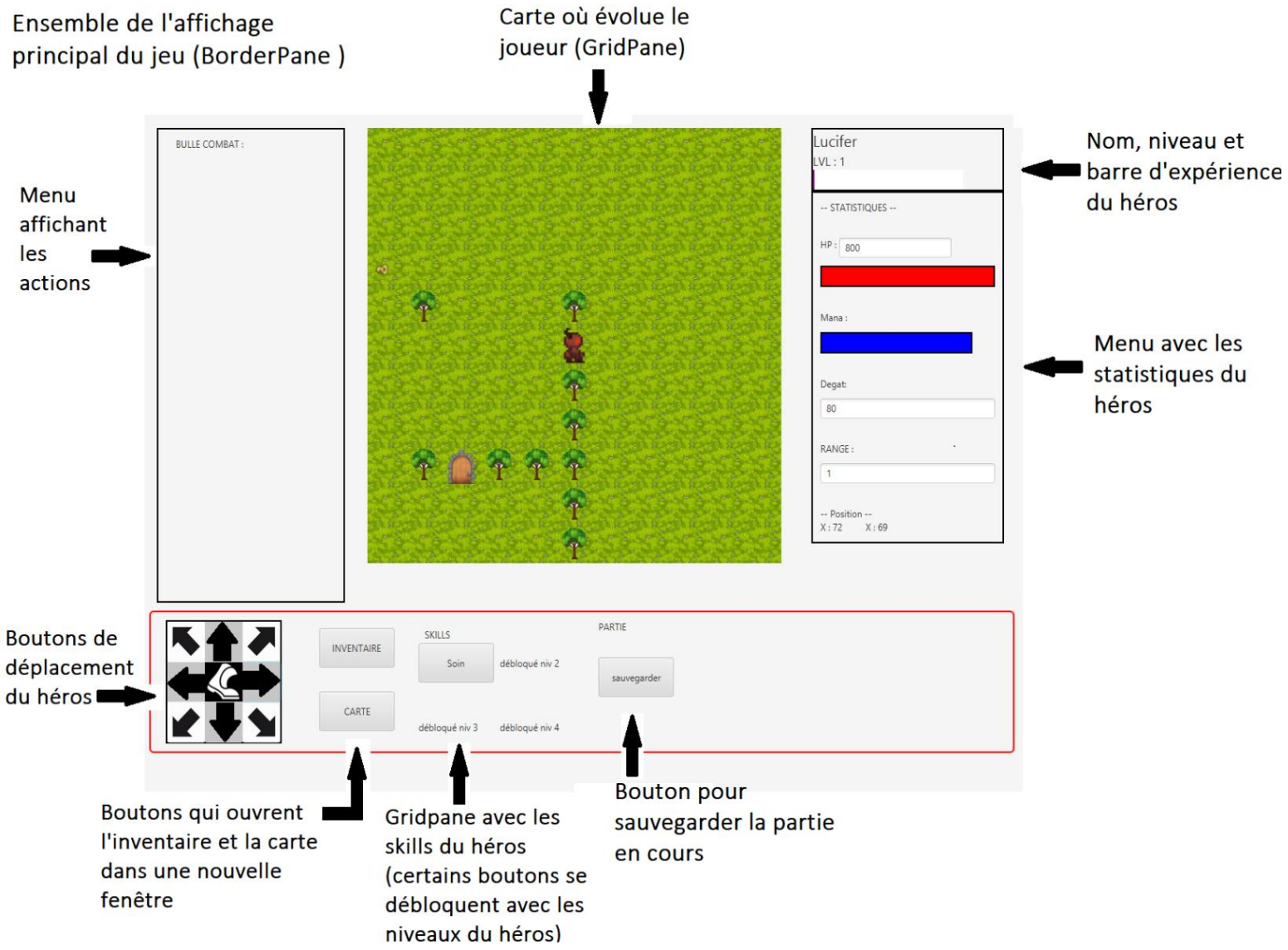
Charger une Partie : Permet a l'utilisateur de charger sa partie qu'il aura au prealable sauvegarder auparavant, il conserve ainsi toute ces donnees precedente, son inventaire, son experience, ces niveau et competences precedemment obtenue :

La presentation de ce Menu n'est pas optimale a mes yeux.

En y ajoutant par exemple une fonctionnalite de connexion avec un nom d'utilisateur et un mot de passe que je relieraisn a notre base de donnees.



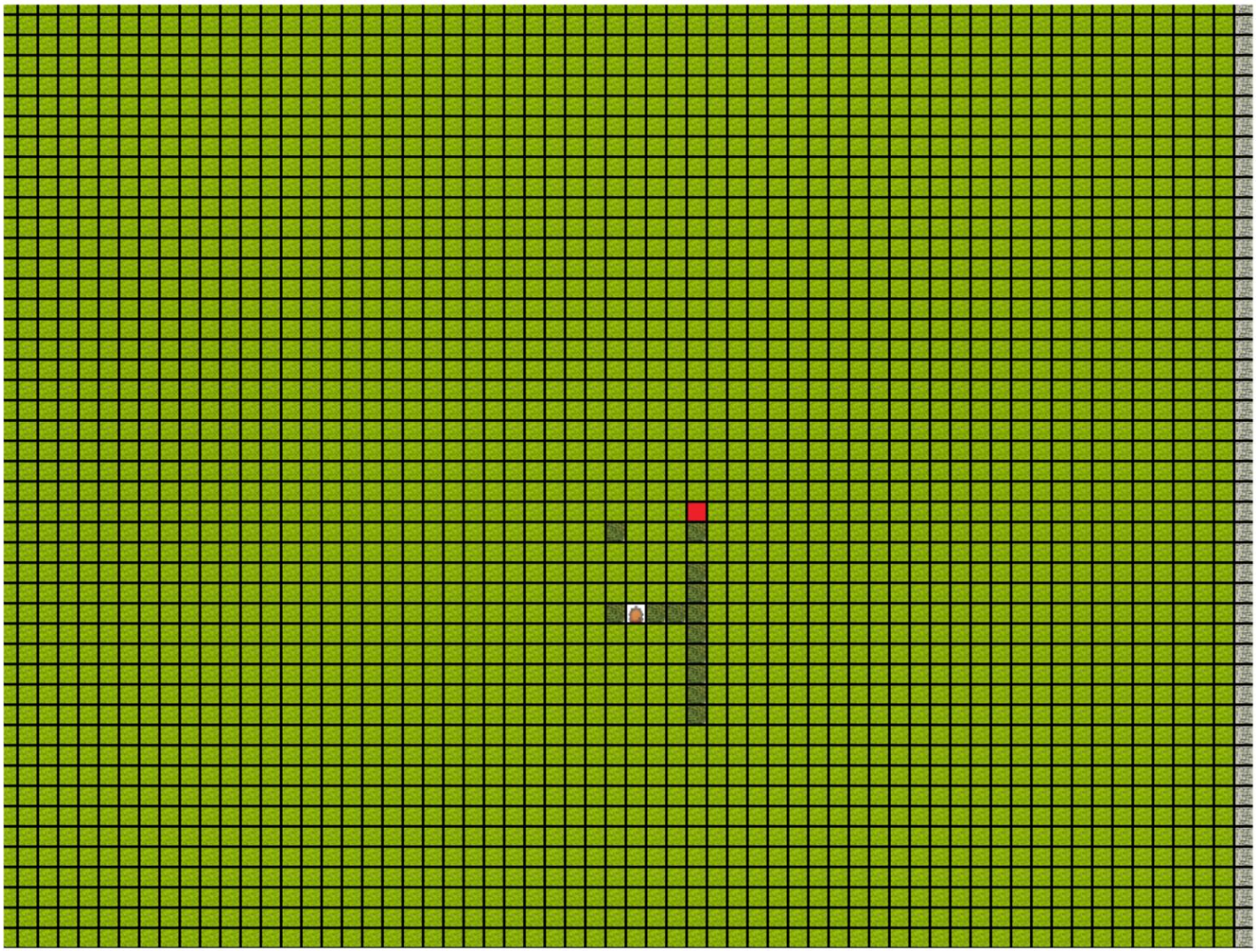
Apercu de l'affichage principal :



Inventaire ouvert avec le bouton inventaire :

L'inventaire est un BorderPane avec « Items » en haut et le stockage en bas, le milieu est compose des objets, chaque objet sera represente par son nom suivi de son image, de sa description puis des boutons Use et Drop (comme ici avec la cle). Le bouton Use sert a utiliser un item et le bouton Drop a en lacher un, il y a un maximum de 5 items dans cet inventaire. L'inventaire se remplira des objets ramasses par le heros.

Apercu de la Map obtenu en appuyant sur le bouton Carte :



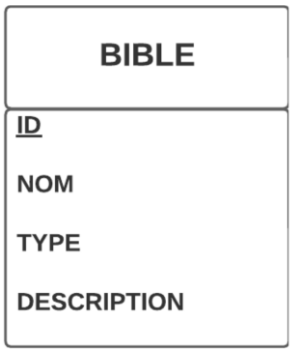
La carte est un GridPane representant la matrice du jeu avec tous ses elements, on reprend la carte de l'affichage principal et on l'affiche en montrant plus de cases. Cette carte est centree sur le heros et affiche tous ce qu'il y a autour de lui mais on peut faire defiler la carte avec la souris si l'on veut voir plus loin.

Modele BD :

Dans notre modèle de base de données je suis alles a l'essentiel, même si j'aurais aimés diviser ma table Bible en plusieurs sous-tables correspondant à l'univers (malheureusement encore trop peu développer) de mon projet, La table Bible répertorie donc, tout Objets, monstres ou Décor interactif avec notre Hero.

La Seconde Table s'intéresse a la partie Sauvegarde de notre jeu, le joueur peut sauvegarder à tout moment sa partie en lui attribuant un NOM dans le champ texte juste en dessous du bouton de Sauvegarde. Il chargera ensuite sa sauvegarde qui a été stocker dans la base de données.

Voici les deux Tables en question :



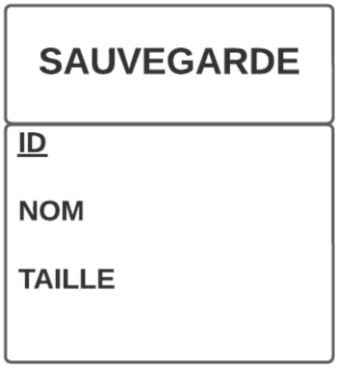
BIBLE :

La colonne ID (INTEGER) est la clé primaire de la classe BIBLE.

La colonne Nom (STRING) correspond au nom de l'item Bible.

La colonne Type (STRING) correspond au type d'objet.

La colonne Description (STRING) correspond à la description de l'objet.



SAUVEGARDE :

La colonne ID (INTEGER) est la clé primaire de la classe SAUVEGARDE.

La colonne nom (STRING) correspond au nom de la sauvegarde.

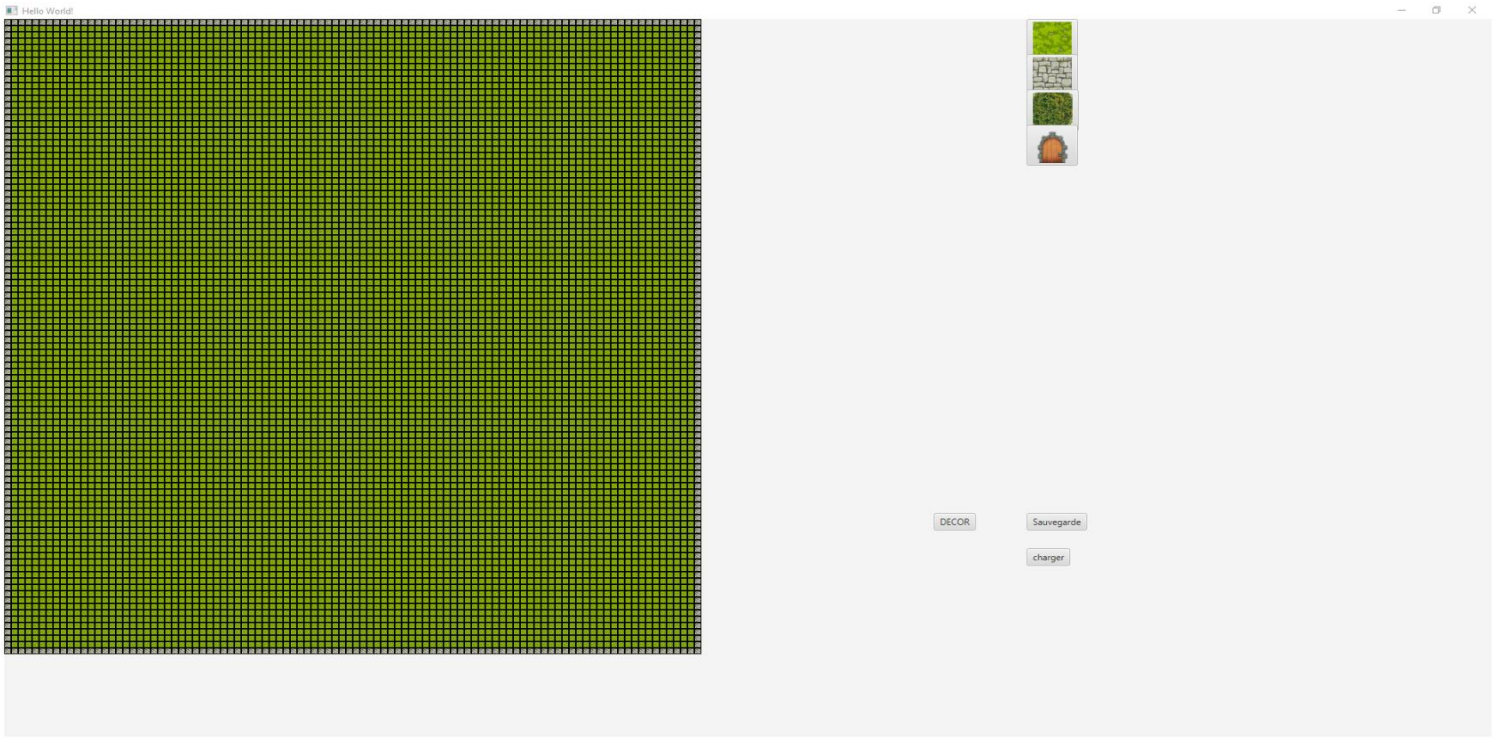
La colonne taille (STRING) correspond à la taille du fichier de sauvegarde.

Lancer/Jouer au jeu :

Pour jouer au jeu, lancer la compilation du projet, on se retrouve devant un menu, en appuyant sur lancer une nouvelle partie, vous pourrez jouer au jeu, il suffit de cliquer sur les flèches directionnelles pour déplacer le héros, vous pouvez ainsi ramasser des objets, tuer des ennemis en se mettant a portée d'eux et progresser dans le niveau. Le héros peut utiliser des skills pour aider la progression. Le joueur peut ouvrir son inventaire (bouton inventaire) pour consulter les objets qu'il a en sa possession, il peut egalement ouvrir la carte (bouton carte) pour voir où se situe le heros et ce qu'il y a aux alentours. Il peut egalement sauvegarder la partie en cours pour pouvoir la reprendre plus tard, pour ce faire il doit appuyer sur « charger une partie » au moment de la compilation du projet. S'il y a des problemes de compilation du projet, cela peut etre lie a la version de la JDK, pour regler ce probleme, il suffit de changer la version de la JDK dans les proprietes du projet.

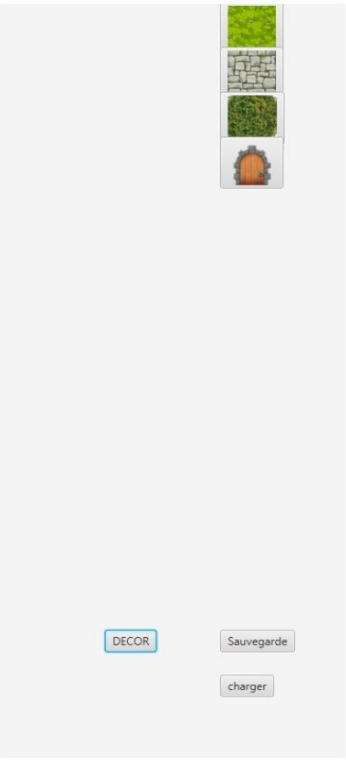
Lancer Creation_Carte :

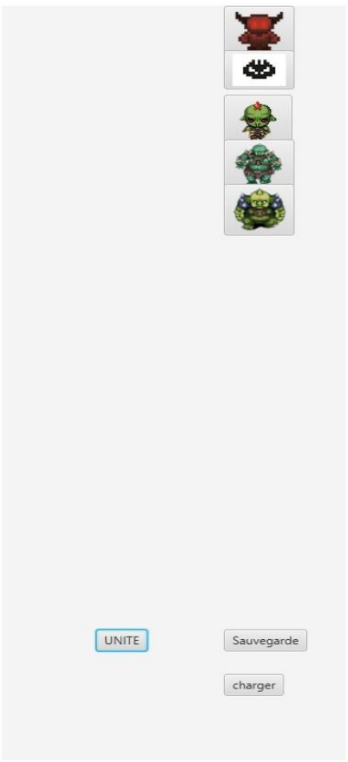
Ceci est une classe avec sa propre methode Main qu'on lance pour pouvoir creer des cartes pour notre jeu diablotin. C'est un outil de developpement (actuellement) car j'ai juge qu'il n'est pas simple d'utilisation pour des joueurs.

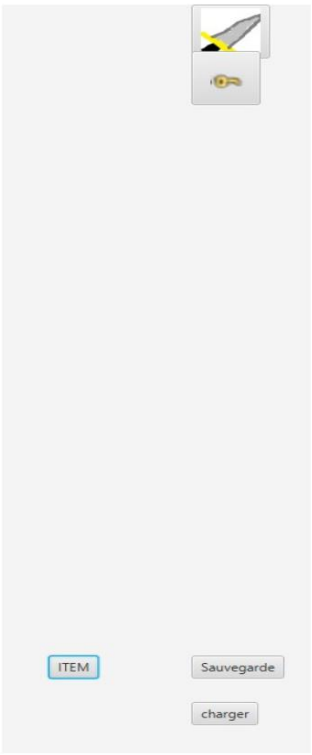


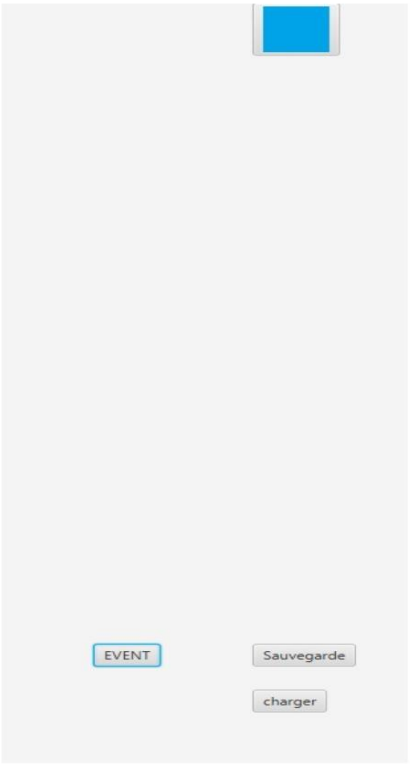
A gauche la carte (vierge), a droite des boutons pour selectionner des objets a placer sur la carte.

Le bouton DECOR qui permet de changer de type d'objet (DECOR→UNIT→ITEM → EVENT) :









Decor - Uniter - Element/Item - Event/Quete

Si on selectionne un decor on peut cliquer/drag sur la carte pour les placers, pour UNIT, ITEM, EVENT on ne peut que cliquer pour ne pas faire de doublons.

Pour EVENT, c'est une zone ou on doit tuer des unites pour gagner une epee donc pour placer la zone, il faut cliquer une premiere fois dans le coin en haut a gauche puis une deuxieme fois dans le coin en bas a droite pour creer une zone de coordonnee du coin en haut a gauche au coin en bas a droite.

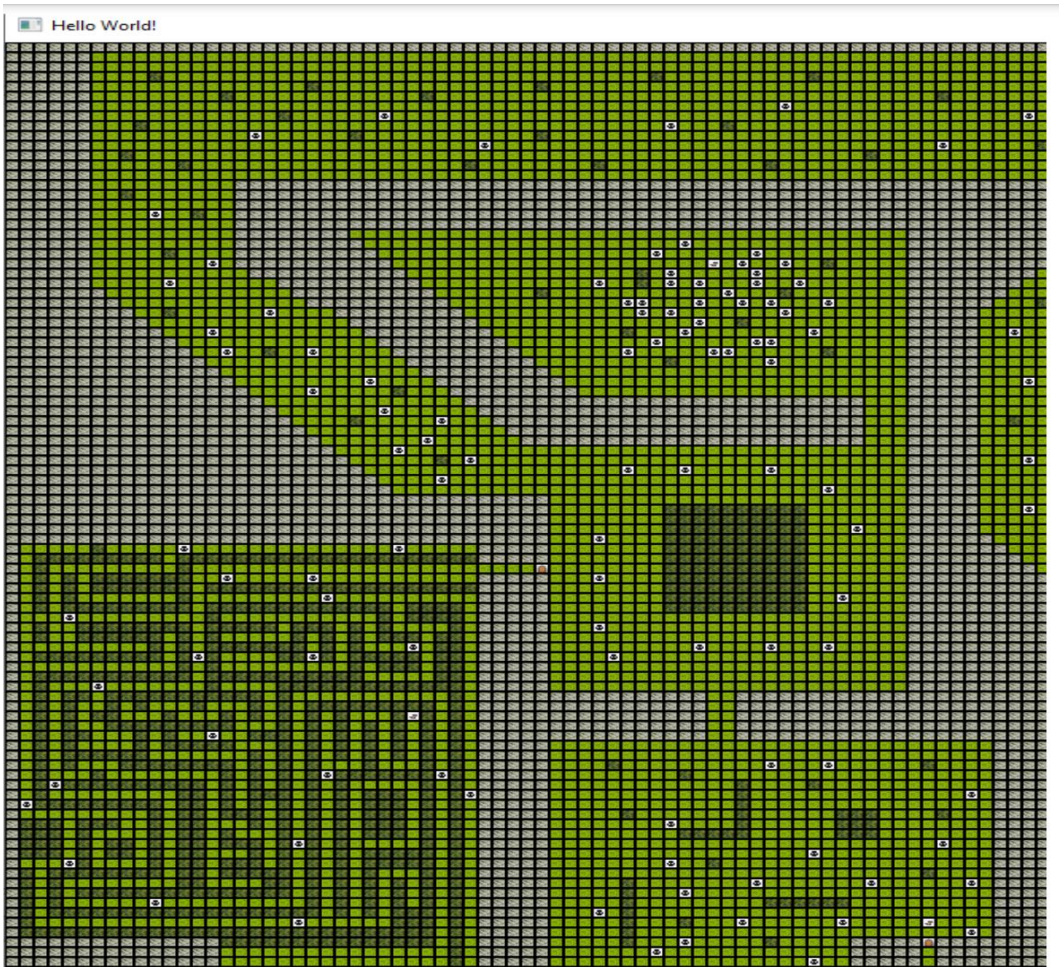
- On peut aussi sauvegarder la carte alors elle sera dans le fichier CREATION sous le nom sauvegarde_creation.txt.
- On peut donc aussi charger une carte mais elle doit etre sauvegarder par creation_carte.java et doit etre dans le fichier CREATION sous le nom sauvegarde_creation.txt.

Le but est de creer une carte ou d'ameliorer une carte deja faite pour notre jeu, apres la carte finis il faudra la mettre dans le dossier NIVEAU et la nomme STAGE1.txt (on peut l'appeler

STAGE2.txt pour faire une suite de niveau mais faudra adapter le code). Pour supprimer une UNIT, ITEMS il faut mettre un DECOR par-dessus.

Contrainte a ne pas faire pour une carte de jeu correct : Il est possible de mettre plusieurs Lucifer ou aucun (notre hero en rouge) mais pour un jeu correct il faut en mettre un seul. (Aucun hero, en posera un quand meme en 70 70)

- On peut poser plusieurs UNITER au meme endroit si on clique plusieurs fois.
- On peut enlever les murs des bordures, mettre des portes sans cles et creer une carte impossible a finir.



Exemple d'une partie cree

Arborescence du Fichier :

Voici L'arborescence de notre fichier de facons a ce que l'utilisateur puisse le consulter sans problemes :

BUILD/ : Ce repertoire contient les fichiers generes lors de la construction du projet, tels que les fichiers compiles, les bibliotheques externes et autres ressources necessaires a l'execution de l'application.

CREATION/ : Ce repertoire contient des fichiers de carte crees. Il peut s'agir de fichiers de donnees ou de fichiers de configuration specifiques a notre application de creation de cartes.

DIST/ : Ce repertoire contient les fichiers distribuables de votre application, tels que les fichiers JAR executables ou les packages d'installation.

map-Test/ : Ce repertoire contient des fichiers de test de cartes. Il peut etre utilise pour tester les fonctionnalites de votre application avec des cartes specifiques.

NIVEAU/ : Ce repertoire contient les niveaux de notre jeu ou de notre application basee sur les cartes. Chaque niveau peut avoir ses propres fichiers et ressources associes.

Sauvegarde/ : Ce repertoire contient les Sauvegarde effectue par le joueur lors d'une partie.

Src/ :

1. **Dossier "Diablo"** : Ce dossier contient le fichier principal de notre application. Il est nomme "Main" ou quelque chose de similaire. Ce fichier contient le point d'entree de votre programme, c'est-a-dire la methode "main" où l'execution de votre application commence.
2. **Dossier "Image"** : Ce dossier contient des fichiers d'images utilises dans notre application. Ces images peuvent etre des icones, des arriere-plans ou tout autre element graphique necessaire a votre jeu.
3. **Dossier "Jeu" :** Ce dossier contient deux autres fichiers importants pour notre jeu :

Fichier "Model" : Ce fichier represente la partie modele de notre application. Il peut contenir des classes et des logiques qui decrivent la structure des objets de votre jeu, les

regles du jeu, la gestion des collisions, etc. Le modele est generalement responsable de la gestion des donnees et de la logique metier.

Fichier "Vue" : Ce fichier represente la partie vue de notre application. Il peut contenir des classes et des logiques qui gerent l'affichage graphique du jeu, l'interaction avec les utilisateurs, les menus, les animations, etc. La vue est generalement responsable de la presentation des donnees au joueur.

Ces fichiers et dossiers constituent une structure de base pour notre projet Java. Cependant, veuillez noter que la structure exacte varie en fonction des conventions etablie entre j'lors de choix de conception specifiques que j'ai faits.

MANUEL D'INSTALLATION :

Assurez-vous d'avoir une machine Java correctement configuree :

1. Verifiez que Java Development Kit (JDK) est installe sur votre machine.
2. Vous pouvez le telecharger depuis le site officiel d'Oracle et suivre les instructions d'installation appropriees pour votre systeme d'exploitation.
3. Placez-vous dans le repertoire où se trouve le fichier .jar de votre projet, normalement dans le dossier "dist".
4. Ouvrez une fenetre de terminal ou de ligne de commande dans ce repertoire.

Utilisez la commande suivante pour executer le fichier .jar de votre projet :

Java -jar Jeu.jar

ou

java --module-path /chemin/vers/javafx/lib --add-modules javafx.controls,javafx.fxml -jar Projet/dist/Jeu.jar

Appuyez sur la touche "Entree" pour executer la commande. Le lancement de l'application **JavaFX** devrait commencer.

Assurez-vous que toutes les dependances requises par votre projet **JavaFX** sont presentes et accessibles a partir du fichier .jar.

Cependant normalement ce projet contient bien les bibliotheques externes et ressources necessaires, assurez-vous sinon de les inclure dans le repertoire approprie et de les referencer correctement dans ce projet.

Probleme Rencontre :

Durant le Projet j'ai rencontré plusieurs problemes en ce qui concerne les parties delicates du developpement du Jeu à savoir les Threads et la Sauvegarde.

Threads

J'ai été confrontés à plusieurs problèmes liés à l'utilisation des threads. Les threads sont des éléments essentiels pour gérer des taches concurrentes et maintenir une interface utilisateur réactive. Cependant, leur utilisation peut parfois poser des défis techniques. Dans ce texte, je vais décrire les problèmes spécifiques rencontré et les solutions que j'ai explorées pour les résoudre.

- Problème de mise à jour de l'interface utilisateur : L'un des problèmes les plus courants rencontrer était lie à la mise à jour de l'interface utilisateur à partir d'un thread autre que le thread de l'interface utilisateur lui-même c'est à dire les Sprites et les Déplacements des Unité. Lorsque j'tentions de mettre à jour des éléments graphiques tels que des images ou des données à partir d'un thread de fond, j'obtenais des exceptions et une interface utilisateur non réactive. Cela était dû au fait que JavaFX exige que toutes les modifications de l'interface utilisateur soient effectuées sur le thread JavaFX principal.

Solution : Pour résoudre ce problème, j'ai utilisé la classe Platform de JavaFX qui fournit des méthodes pour exécuter du code sur le thread JavaFX principal. J'ai enveloppé mes modifications d'interface utilisateur dans des appels à Platform.runLater(() -> {/* code de mise à jour ici */}), ce qui garantit que les modifications sont effectuées sur le thread principal.

- Problème de synchronisation : Un autre problème courant était lie à la synchronisation des threads lorsqu'ils accédaient a des ressources partagées. Lorsque plusieurs threads tentaient d'accéder ou de modifier une même ressource en même temps, des problèmes de concurrence se produisaient, tels que des conditions de course ou des accès concurrents non désirés.

Solution : Pour éviter les problèmes de synchronisation, j'ai utilisé des mécanismes de verrouillage tels que les instances statiques ou les classes du package java.util.concurrent. Ces mécanismes m'ont permis de coordonner l'accès aux ressources partagées entre les threads, assurant ainsi une exécution sûre et cohérente de mon application.

- Problème de gestion des taches longues : Dans mon projet, j'ai également dû gérer des taches longues qui risquaient de bloquer l'interface utilisateur si elles étaient exécutées sur le thread JavaFX principal. Par exemple, lors de l'execution d'operations de calcul intensif des déplacements et des attaques de chaque unité ou de l'accès à des ressources externes, l'interface utilisateur pouvait devenir non réactive, ce qui était une expérience utilisateur médiocre.

Solution : Pour résoudre ce problème, j'ai utilisé des threads de fond pour exécuter les taches longues. J'ai utilisé la classe JavaFX Task que j'ai regrouper dans une Sous-classe Service appeler ServiceGame dans notre projet, qui permet d'exécuter des taches asynchrones et de mettre à jour l'interface utilisateur en cours d'exécution. J'ai également utilise des indicateurs visuels tels que la barre de Mana qui se remplit d'elle-même au cours du temps, gérer dans Fill_Mana.

Sauvegarde

Dans ce projet, j'ai été confrontés à des défis lors de l'implémentation de la fonctionnalité de sauvegarde. La sauvegarde est une étape essentielle pour préserver les données de l'application et permettre aux utilisateurs de retrouver leur partie ultérieurement mais aussi lors de la

création de Map afin d'éviter toute fonction de création Superflue à mesyeux. Cependant, la gestion de la sauvegarde peut être complexe en raison de divers facteurs, tels que le format des données à sauvegarder et les interactions avec les composants de l'interface, les statistiques de notre Hero. Dans ce texte, j'vais expliquer les problèmes spécifiques que j'ai rencontrés et les solutions que envisagées pour les résoudre.

- Problème du format de sauvegarde : L'un des principaux problèmes que j'ai rencontrés était de déterminer le format de sauvegarde à utiliser pour mes données. JavaFX offre plusieurs options, notamment la sérialisation des objets, l'utilisation de formats de fichiers tels que JSON ou XML, ou encore la persistance dans une base de données. Chaque option a ses avantages et ses inconvénients, et il était crucial de choisir celle qui correspondait le mieux à la structure de mes données et aux besoins de mon application.

Solution : Apres avoir évalué les différentes options, j'ai opté pour la sérialisation des objets en utilisant la classe Java Serializable. Cela m'a permis de convertir facilement des objets JavaFX en un format binaire et de les enregistrer dans un fichier. Cependant, j'ai également pris en compte les implications de la sérialisation, telles que la nécessité de gérer les versions des objets sérialisés pour éviter les problèmes de compatibilité a l'avenir.

- Problème de gestion des évènements de sauvegarde : Un autre problème que j'ai rencontré était de déterminer quand et comment déclencher la sauvegarde des données. Dans une application JavaFX, il peut y avoir plusieurs évènements qui nécessitent une sauvegarde, tels que la modification des données par l'utilisateur, la fermeture de l'application ou l'appui sur un bouton de sauvegarde.

Solution : Pour résoudre ce problème, j'ai utilisé une combinaison de différentes techniques. J'ai attaché des écouteurs d'évènements aux composants de l'interface utilisateur pour détecter les modifications de sauvegarde lorsque nécessaire, dans le fait de charger une partie plus précisément. J'n'ai malheureusement pas utilise des méthodes de rappel (callbacks) pour gérer les évènements de fermeture de l'application ou des actions utilisateur spécifiques mais j'y pensions. Enfin, j'ai ajoute un bouton de sauvegarde explicite, ainsi qu'un Textfield préremplis pour nommer la sauvegarde, pour permettre aux utilisateurs de contrôler manuellement le processus de sauvegarde.

- Problème de traitement des erreurs de sauvegarde : Un autre défi était de gérer les erreurs qui pouvaient survenir lors de la sauvegarde des données. Des problèmes d'instance lie au Thread pouvaient compromettre la sauvegarde.

Solution : Pour faire face à ces problèmes, j'ai opté pour utiliser des mécanismes de sauvegarde incrémentielle ou de sauvegarde différée pour garantir que les données ne soient pas perdues même en cas d'échec de la sauvegarde initiale.

Conclusion :

Cette experience de projet a ete a la fois eprouvante et initiatique, en plongeant dans un defi de grande envergure. En travaillant j'ai fait face a des defis techniques, des problemes de coordination et des periodes de stress. Cependant, cette experience m'a egalement permis de developper certaines de mescompetences, de repousser un peu plus meslimites et d'explorer de nouvelles perspectives.

Pourtant, au milieu de cette intensite de travail, j'ai realise que j'ai neglige une partie primordiale de notre projet : la presentation et le rapport a rendre. En me concentrant principalement sur le développement technique, j'ai sous-estimé l'importance de communiquer efficacement mes idees, mes decisions et mes realisations.

La presentation et le rapport sont des elements cles d'un projet, car ils permettent de documenter notre travail, de le presenter de maniere claire et concise, et de mettre en valeur les realisations de l'equipe. Cela inclut la description des choix de conception, l'explication des fonctionnalites majeures, l'analyse des defis rencontres et des solutions mises en place, ainsi que la demonstration des resultats obtenus.

En realisant cette negligence, j'ai compris que la communication efficace est une competence essentielle dans le domaine de l'informatique. Il ne suffit pas seulement de bien coder et de mettre en place les fonctionnalites, mais il est egalement crucial de pouvoir expliquer notre travail, mesprocessus et mesresultats aux autres membres d'une equipe, d'un superviseur me permettant de progresser, mais aussi dans un cadre professionnel, c'est à dire, aux clients potentiels ou aux utilisateurs finaux.

Cependant, j'ai egalement appris que meme si j'ai neglige cette partie du projet, il est et reste de notre devoir de la rectifier. J'ai decide de consacrer du temps supplementaire a la redaction d'un rapport detaille. En esperant que malgre tout, cela permettra de fournir aux autres une comprehension complete de ce projet.

Cette experience m'a donc rappele l'importance de la communication et de la documentation dans le processus de developpement de logiciels. J'ai realise que la presentation et le rapport ne sont pas des taches secondaires, mais des elements essentiels pour partager notre expertise, apprendre des autres et renforcer notre crédibilité.

En fin de compte, cette expérience m'a permis de prendre conscience de l'importance de la presentation et du rapport dans un projet, et je m'engage à accorder à ces elements la priorite qu'ils meritent a l'avenir. Je suis déterminé à apprendre de mes erreurs et a utiliser cette experience pour grandir tant professionnellement, que pour mes competence et polyvalence a travailler, etre capables de mener a bien des projets de grande envergure tout en communiquant efficacement notre travail et mesrealisations.

Bibliographie :

L'essentiel de mes informations vien avant tout de mes cours de POO-IHM, le reste étant pour des aspect plus technique tel que la base de donnée et sinon les sprites utilisé pour le visuel.

https://terminalroot.com/how-to-connect-to-sqlite-with-java/

Pour plus d'information : https://github.com/xerial/sqlite-jdbc

Jun 16, 2022, Taro L. Saito

Sprites and Animation - How to Make a 2D Game in Java #3

12 oct. 2021 ,RyiSnow, Youtube

https://elthen.itch.io/2d-pixel-art

Site de Sprite gratuit en Ligne

RESUMER :

Arthur Haye, inscrits à l'université de Poitiers. Dans le cadre de mes études, j'ai entrepris un projet durant le mois de mai : développer un jeu de type Hack 'n'Slash, intitulé "Diablotin".

"Diablotin" est entièrement codé en Java et s'inspire de la célèbre série de jeux "Diablo". Dans ce jeu, vous serez plongé dans un univers fantastique où vous devrez affronter des hordes d'adversaires toujours plus nombreux. Votre objectif principal sera de trouver des équipements puissants et de débloquer des pouvoirs mystérieux pour améliorer vos compétences de combat.

Le gameplay de "Diablotin" se veut fluide et dynamique, offrant action et une expérience palpitante à chaque instant. Au fur et à mesure de votre progression, vous rencontrerez des ennemis de plus en plus redoutables et des boss terrifiants qui mettront vos talents à rude épreuve. Vous devrez vous frayer un chemin à travers des donjons complexe, tout en restant vigilant face aux pièges et aux monstres qui se dresseront sur votre chemin.

L'exploration joue également un rôle clé dans "Diablotin". Vous découvrirez des trésors cachés, des pieges et des quetes annexes qui vous récompenseront avec des objets rares. Votre ultime défi sera de terrasser le boss final, une créature démoniaque d'une puissance inouïe. Une fois cette menace éliminée, vous pourrez enfin vous échapper de ce monde hostile et atteindre la victoire tant méritée.

Plongez dans l'aventure épique de "Diablotin" et profiter de cette expérience Hack 'n'Slash !

Ça me parait bien.

Arthur Haye, enrolled at the University of Poitiers. As part of i studies, i undertook a project during the month of May: develop a Hack 'n'Slash type game, entitled "Diablotin".

"Diablotin" is entirely coded in Java and is inspired by the famous series of "Diablo" games. In this game, you will be immersed in a fantastic universe where you will have to face ever-increasing hordes of opponents. Your main objective will be to find powerful equipment and unlock mysterious powers to improve your combat skills.

The gameplay of "Diablotin" is intended to be fluid and dynamic, offering action and a thrilling experience at all times. As you progress, you will encounter increasingly formidable enemies and terrifying bosses that will put your skills to the test. You will have to fight your way through complex dungeons, while remaining vigilant against the traps and monsters that will stand in your way.
Exploration also plays a key role in "Diablotin". You will discover hidden treasures, traps and side quests that will reward you with rare items. Your ultimate challenge will be to defeat the final boss, a demonic creature of incredible power. Once this threat is eliminated, you will finally be able to escape from this hostile world and achieve the much-deserved victory.

Dive into the epic adventure of "Diablotin" and enjoy this Hack 'n' Slash experience !

Sounds good no ?