RAPPORT DE PROJET

Soline Corm

Vladislas De Haldat Du Lys

Kevin Garnier

Charly Martin Avila

Ana Parres Cerezo

Reproduction du jeu « Wargroove » en Java

Introduction

WarGroove est un jeu vidéo de stratégie au tour par tour dont le but est de vaincre l'armée ennemie soit en éliminant le commandant ennemi, soit en détruisant sa base principale.

Figure 1 Exemple de plateau de jeu

Les joueurs doivent prendre possession d'endroits stratégiques : les bases de recrutement qui servent à obtenir des troupes, ou des villages pour augmenter leur revenu par tour, ce qui leur permet d'acheter des troupes. Les terrains possèdent des caractéristiques différentes, apportant un bonus ou un malus sur la défense ou pénalisant le déplacement d'une troupe. Par exemple, il est plus facile pour les troupes de marcher sur un chemin qu'à l'intérieur d'une forêt mais la forêt offre une meilleur protection que le chemin.

La stratégie repose également pour les joueurs dans le fait de déterminer quelles unités doivent s'affronter, et tenir compte des différents rapports de forces entre les types de troupes. Par exemple, si un joueur attaque un soldat ennemi avec un villageois, ce dernier ne fera aucun dégâts et le soldat le vaincra. Par contre, si le joueur attaque ce soldat avec un géant, son attaque sera beaucoup plus efficace. Il faut prendre en compte une possible représaille de la troupe attaquée qui pourrait sévèrement blesser l'attaquant.

L'objectif de ce projet est de reproduire les règles du jeu.

1 Dossier de spécification

1.1. Cas d'utilisation

Lors du lancement du programme, l'utilisateur arrivera sur le menu principal du jeu.
Si l'utilisateur choisit "Setting", il accédera aux paramètres du jeu et il pourra changer la vitesse de la caméra et de son zoom, mettre le jeu en plein écran ou modifier le volume des sons et des musiques.
D'autre part, si l'utilisateur lance une partie, il devra choisir le nombre de joueurs, puis un ensemble de cartes lui seront proposées (chaque carte à une nombre spécifique de joueurs). Il y aura un affichage de la carte présélectionnée, pour faciliter le choix de l'utilisateur. Ensuite, il faudra sélectionner un biome ce qui changera l'aperçu de la carte et la musique d'ambiance. Il peut également choisir un pourcentage de revenu par tour qui va influencer la somme initiale et le revenu généré par tour. Ainsi un pourcentage élevé aura tendance à simplifier le jeu, car les joueurs auront des troupes illimités. A contrario, un pourcentage faible oblige le joueur à être parcimonieux sur les troupes qu'il achète.
Une fois la partie lancée, l'utilisateur aura toujours le choix d'accéder à un menu, ce qui lui permet de modifier les paramètres, de retourner sur le menu principal ou de quitter le jeu.

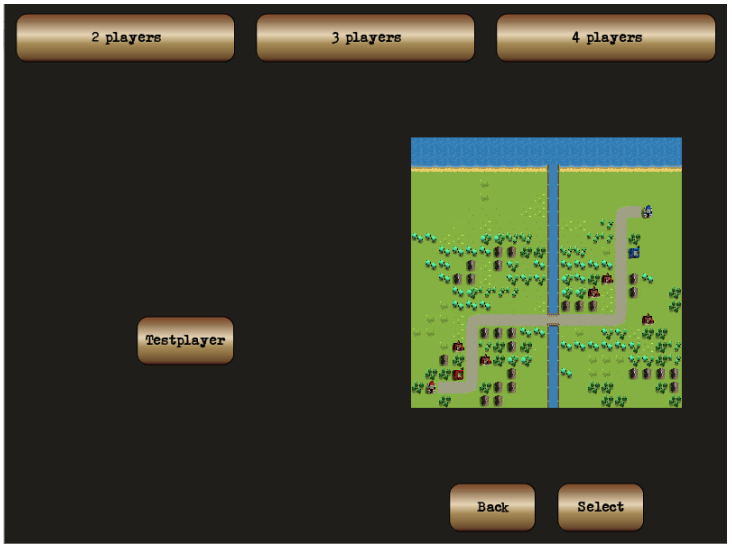


Figure 2 Scénario des premières manipulations

1.2. Scénario du jeu

Quand le jeu commence, chaque joueur possède un commandant, un quartier général ainsi qu'une somme d'argent initiale qui lui permettra d'acheter d'autres troupes dans les casernes de recrutement. Cette structure est d'une importance capitale, sans elle, le joueur est voué à perdre. Il s'agit donc d'un point stratégique à défendre aussi bien que la base principale.
Le déplacement des personnages joue un rôle majeur dans Wargroove , certaines troupes sont plus rapides ou plus puissantes que d'autres, alors selon la stratégie employée, les joueurs doivent choisir s'ils veulent attaquer avec une troupe, la déplacer, ou la faire attendre. Cependant pour agrandir leur armées, les joueurs ont besoin d'argent, d'où l'intérêt de capturer des villages. En effet, chaque village capturé augmente les revenus du joueur par tour. Les joueurs peuvent capturer un village détenu par un autre joueur en le détruisant, il devient alors neutre. Dans cet état, le village n'appartient à personne et le joueur peut le capturer instantanément en envoyant une troupe l'attaquer. Ceci est aussi valable pour les casernes de recrutement. Les quartiers généraux sont les seules structures qui peuvent être définitivement détruites, ce qui implique l'élimination immédiate du joueur propriétaire. De même, si le commandant d'un joueur est éliminé, celui-ci perd. Quand un joueur a perdu, toutes ses troupes sont supprimées et ses structures libérées.
La partie se termine lorsqu'il ne reste qu'un seul joueur.

2 Dossier de conception

2.1. Cahier de charges :

2.1.1. Objectifs principaux :

Nous nous sommes fixé comme objectif principal d'implémenter les règles de base du jeu, c'est à dire les fonctionnalités suivantes :

- Les unités ont des points de vie et une force d'attaque. Le portée d'attaque n'est pas forcément la même en fonction du type de troupe
- Chaque fois qu'on essaye de déplacer un personnage, indiquer au joueur quels mouvements sont admissibles.
- On peut voir quelles unités ont déjà joué.
- Les tours sont bien définis et le jeu s'arrête quand une des deux armées est vaincue.

2.1.2. Objectifs secondaires :

- Ajout de cartes créées manuellement
- Créer nous même la musique et les dessins de l'interface graphique

2.2. Diagrammes des classes (voir détails Annexe) :

Figure 3 : Diagramme des classes illustrant les vues

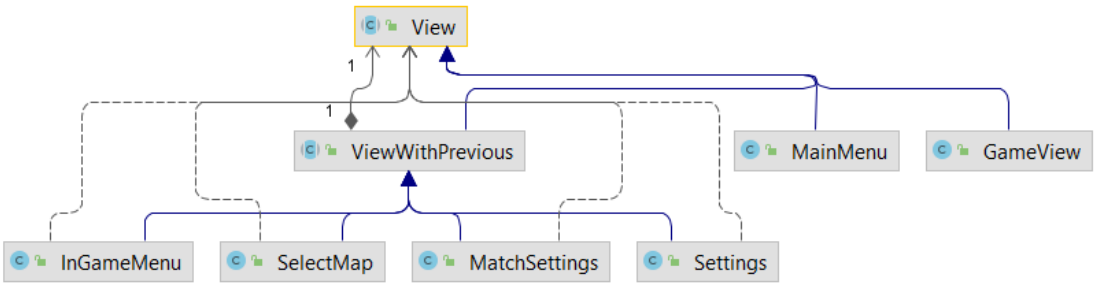


Figure 4: Diagramme des classes illustrant l'écran de configuration d'une partie

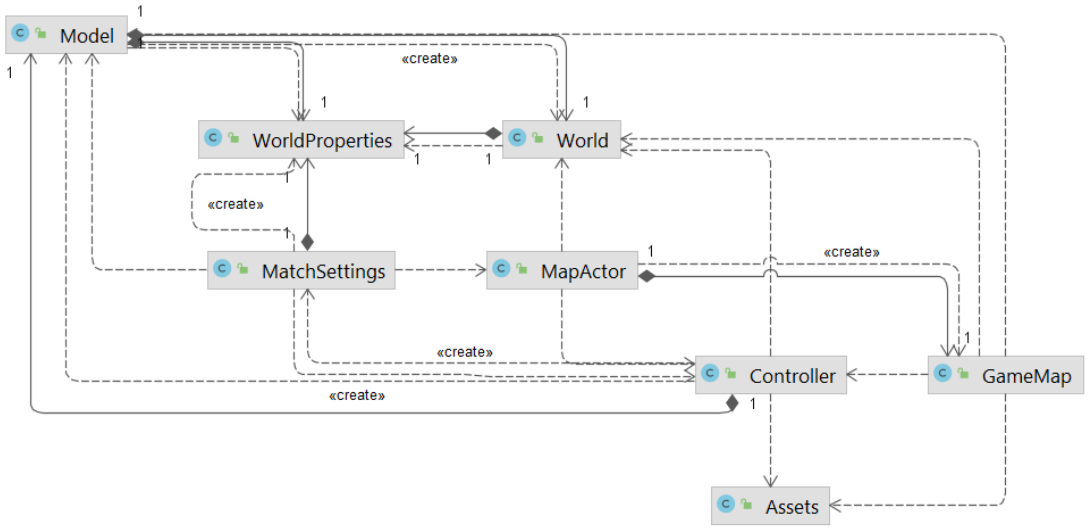


Figure 5: Diagramme illustrant l'écran paramètres

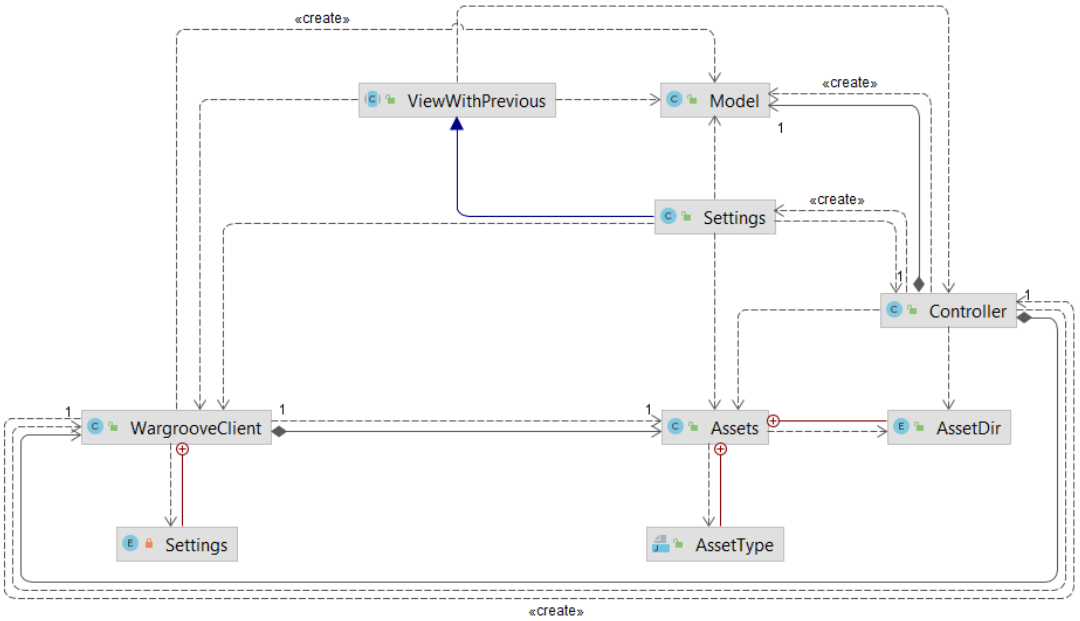
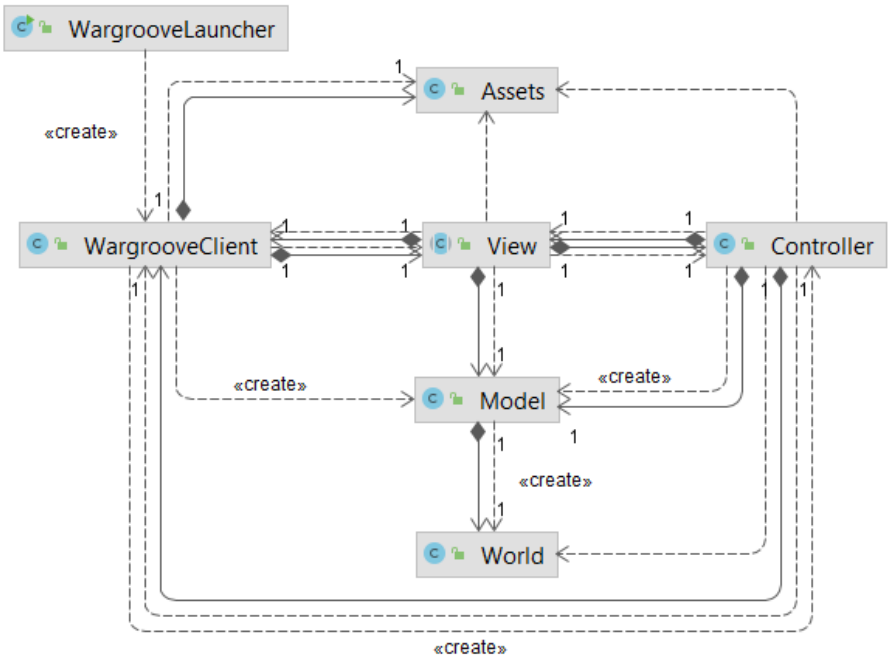


Figure 6 Diagramme des classes illustrant le paterne model-View-Controller



3 Résultats

Catégorie	Objectifs	Implementé	À futur	Commentaires	Développeur
Plateau du jeu	Créer différents biomes	X		Grass, Ice, Desert, Volcano	Vladislas
	Créer différents type de terrains	X		Chemin, forêt, montagne, mer, plage…	Vladislas
	Dessins des biomes	X		Sur ice, l'eau des rivières est congelée et on peut marcher sur elle.	Kevin
	Effets sur déplacement	X		Différents coût de déplacement selon les terrains	Vladislas
	Création des différentes maps	X		Manuellement avec Tiled	Kevin, Ana, Soline
Personnages du jeu	Damage Matrix	X		Permet de calculer les dégâts que fait un personnage à un autre	Charly
	Dessins des personnages	X			Ana
	Créer différents type de personnages[1]	X		Commander, Soldier, Archer, Mage, Spearman…	Charly
Menus	Menu d'accueil	X		Permet d'accéder aux autres menus ou de quitter l'application	Kevin et Soline
	Menu des paramètres	X		Taille de l'écran, sons	Kevin et Soline
	Menu en jeu	X		Permet de mettre en pause le jeu et de régler des paramètres d'écran et son. Ainsi que de retourner sur le menu principal et de quitter le jeu.	Kevin
Mécaniques du jeu	Déplacement de troupes	X		Une troupe peut transpercer une unité de la même faction	Kevin
	Mise en attente de troupes	X			Kevin
	Attaque de troupes à d'autres troupes	X		L'attaque est toujours en horizontal ou en vertical, jamais en diagonal	Ana et Charly

[1]On voulait ajouter d'autres types de personnages qui seraient plus adaptés aux milieux aériennes ou aquatiques (amphibiens, aéronautes, dragon)

	Attaque de troupes à structures (libération)	X		Attaquer la structure d'un ennemie et la rendre libre	Ana
	Attaque de troupes à structures (capture)	X		Si la structure est libre, elle appartient au joueur qui la détruit	Ana
	Achat de troupes	X			Kevin
	Contre-attaque des troupes et structures	X		Lorsqu'une entité est attaquée, elle lance une contre attaque si son attaquant est dans son rang d'attaque	Ana
	Système tour à tour	X		Avec l'ajout d'agent et la régénération des structures après chaque tour	Kevin
	Blocage d'une troupe après avoir fait une action	X		Se manifeste en donnant au personnages une couleur grise	Kevin
	Recherche des positions disponibles pour déplacement	X		Se manifeste avec l'affichage de cases marron	Vladislas
	Recherche d' entités attaquables	X		Se manifeste avec l'affichage de cases rouges	Kevin et Vladislas
	Fleche guide	X		Elle permet de créer à l'utilisateur le chemin que le personnage doit parcourir	Kevin
	Système de troupe suivante	X		Permet au joueur de parcourir ses troupes rapidement	Vladislas
	Brouillard		X	Restriction de visibilité du joueur	~
	Systeme de climat		X	Faire varier les conditions climatiques qui ont des effets sur les personnages	~
Views	Preview de la carte choisi	X		Changement de biome visible	Kevin
	Possibilité de zoomer et dézoomer	X		Le zoom est borné, ce qui empêche à l'utilisateur de transposer la carte	Kevin
	Déplacement de la caméra	X			Kevin
Gestion du son	Bande originale	X		Chaque biome a son propre thème	Soline et Vladislas
	Son[2]	X		Bruit sur boutons	Soline

[2] On voulait aussi ajouter des sons aux personnages qui accompagnent leurs mouvements, attaques ou défaites.

Amélioration de l'expérience du joueur	Affichage des données du joueur et de la partie en cours	X		Quantité d'argent, numero de tours…	Kevin
	Mini Zoom sur une tuile	X		En bas à droite, on peut voir une réplique de la tuile sur laquelle survole la souris. Cela est utile quand les cartes sont très grandes.	Kevin
	Affichage si entité est blessée	X		Couleur rouge clignotant	Ana
	Animations des personnages			Déplacement, attaque, K.O.	Ana
	Affichage des points de vie des entités	X		Seulement les dizaines inférieures, si la vie est à 100% pas d'affichage. De 80-99% affiche 9. 1-10% affiche 1	Ana
	Affichage des caractéristiques du personnages pendant l'achat	X			Kevin
	Affichage de zone de danger	X		En noir (même couleur que sur le jeu original) dès qu'on click sur un personnage ennemi pendant notre tour	Kevin
	Codex	X		Montre les caractéristiques du jeu	Ana
	Tutoriel avec mise en place d'un scénario		X	Permettant au joueur de s'habituer au mécanique et de faire une immersion dans le jeu	~
Settings	Plein écran	X			Kévin
	Volume de la musique	X			
	Volume des sons	X			
	Vitesse du zoom	X			
	Vitesse de la caméra	X			
	Sauvegarde des préférences	X		Si on met le mode ecran complet, et on relance le jeu, le jeu sera en mode écran complet	
Multijoueur	Jusqu'à 4 joueurs	X			Kevin
	Serveur		X		~
	IA		X		~
	Plusieurs écrans		X		~

Chargement de fichier	Charger tous les fichier au début du jeu	X		AssetManager, utilisé pour les textures, musiques et descriptions	Kevin

3.1. Éléments à réalisés dans le futur:

Ce sont ceux qui ne sont pas réalisés mais que nous avions déjà pensé à ajouter depuis le début et qu'on voudrait continuer à développer dans le futur. Parmis lesquels se trouvent les éléments cochés comme à futur et les suivants:

- Création de cartes:

L'idée aurait été de s'inspirer du logiciel Tiled, en proposant à l'utilisateur les tuiles dans une liste cliquable. Lorsqu'une tuile est sélectionnée, l'utilisateur peut alors "peindre" la carte. L'utilisateur doit choisir au préalable un biome pour les textures, la taille de la carte dans un intervalle fermé, par exemple [10 ,96] ainsi que le nombre de joueurs et leurs factions. Une fois la carte construite, il ne reste plus qu'à la sauvegarder dans la bonne collection.

- Multiplateforme:

Libgdx est compatible sur Android, IOS, Ordinateurs et Navigateur Web avec un portage HTML. Ainsi, la plupart de ses fonctions sont pensées pour toutes les plateformes. Par exemple, les fonctions des inputListners ont en argument le nombre de doigt et le bouton cliqué. Il y a aussi toute une gestion de l'écran tactile ainsi qu'une gestion du clavier. Le code du jeu qui est commun à toutes les plateformes doit se trouver dans le sous-projet core et le code spécifique aux plateformes doit se trouver dans les sous-projets spécifiques de ces plateformes. Ainsi pour faire le portage Android, il nous aurait suffit de créer un sous projet Android, de créer un launcher Android similaire à la version ordinateur. Il aurait fallu néanmoins changer l'emplacement des assets dans le sous-projet android et créer un manifeste, cela étant dû aux contraintes du système. Lors de la création de la distribution, il faudrait donc copier les assets sur les autres versions.

- Outils de développement:

Nous avons aussi eu l'idée de faire un plugin de maintien de la base de données, qui eût été très pratique pour que le développeur interagisse directement avec la base de données. En effet, au cours du développement, nous nous sommes rendu compte qu'il était obligatoire de réinitialiser la base de données à chaque changement que ce soit au niveau des textures, de la sauvegarde ou du chargement. Souvent, une simple mise à jour de la base de données qui consiste à exporter et importer les cartes, pouvait suffire à compléter les données manquantes qui sont calculées lors de l'importation. Concrètement, le plugin se comporte comme SQL en reconnaissant des mots clés comme UPDATE, DROP collection, ADD collection, LIST collection pour mettre à jour la base de données, supprimer une collection, ajouter une collection et lister les cartes dans une collection.

4 Attitude et choix

4.1. Attitude :

Nous avions une immense quantité d'idées à implémenter, mise à part les objectifs principaux. Nous voulions créer un jeu professionnel, personnel et disponible sur plusieurs appareils. Face à l'énorme quantité de travail, nous avons décidé de diviser les tâches plus complexes et de les faire seul ou dans des sous-groupes de deux personnes.

Nous avions globalement deux équipes: une équipe front-end qui s'occupait de l'interface graphique et des interactions avec l'utilisateur (Kevin, Ana et Soline) et une équipe backend qui s'occupait de gérer le modèle et de fournir une bibliothèque de fonctions à l'autre équipe (Vladislas et Charly).

4.1.1 Méthode Scrum et communication:

Les membres d'un sous groupe étaient constamment en communication, pour se tenir au courant de ses avancées ou pour s'entraider. Ils travaillaient dans une branche à part, qu'ils devaient ajouter postérieurement. Pour communiquer en dehors de la fac nous avons utilisé un groupe Discord.

Les réunions hebdomadaires avec nous permettaient de connaître ce que les autres membres du groupe avaient fait de leur côté, définir les objectifs pour la semaine suivante ou reprendre des objectifs que nous n'avions pas réussi à finir. Nous fixions la date de la réunion comme date limite pour rassembler toutes les versions des différentes branches dans la branche principale. En outre des réunions avec M. Roman-Calvo, nous avons fréquemment organisé des réunions sur Discord.

Néanmoins, il est vrai que nous avions eu quelques difficultés à communiquer en dehors des ces réunions.

4.1.2 GitLab:

Le gros des tâches à faire étant déjà définies, l'utilisation du Board de GitLab était convenablement utile pour faire un suivi de progression du projet, et nous pouvions ajouter de nouvelles tâches qui seraient réalisées plus tard dès que nous avions fini de faire nos tâches. Nous avons vu fleurir sur le Board une myriade d'issues qui n'ont malheureusement pas pu être implémentées dans le temps imparti pour faire le projet.

4.2. Choix :

On nous a offert une grande liberté tout au long du projet, ce qui nous à permis d'utiliser des ressources spécifiques au jeux vidéo ainsi que de personnaliser le jeu et de créer nous même les ressources.

4.2.1. Choix techniques :

Nous avons utilisé Gradle afin de construire notre projet et de gérer les dépendances avec les librairies tierces. Nous avons préféré utiliser Kotlin Gradle par rapport à Maven ou Ants car nous

étions plus habitués à l'utiliser, ce qui nous a permis de mieux personnaliser notre projet en fonction de nos besoins.

Pour réaliser l'interface graphique, nous avons décidé d'utiliser le framework libgdx. Le framework est réputé dans la création de jeux vidéo en Java, en particulier pour les jeux 2D. Il était donc relativement facile de trouver des réponses à nos questions sur internet. De plus, il est très bien documenté, possède un tutoriel ainsi que des exemples de jeu sur le site officiel.

Le framework étant très complet, cela nous a permis de gagner beaucoup de temps sur l'affichage. En effet, il prend en charge, entre autres, les coordonnées dites "de monde" qu'il convertit en coordonnées sur l'écran, il fournit des viewports qui s'adaptent à la taille de l'écran et qui adaptent la taille des différents acteurs. Grâce à cela, nous avons pu porter notre attention sur le contenu de l'interface graphique afin de garantir une expérience utilisateur agréable.

Néanmoins, étant donné que nous ne connaissions pas le framework, l'utilisation de libgdx a compliqué notre travail pendant les premières semaines. Mais une fois habitués, il est devenu très plaisant de travailler avec et nous pûmes faire tellement de choses que notre seule limite fut notre imagination.

Le framework gère aussi l'utilisation des assets dans un AssetManager, où toutes les textures, sons et musiques sont recensés dans une table de hachage. Cela nous a permis de vraiment travailler sur cette gestion de textures afin de limiter la création d'objets similaires, d'éviter de potentielles fuites de mémoire dûes aux assets qui sont chargés sur la RAM.

Libgdx étant connu des codeurs JAVA, il existe plusieurs applications compatibles pour ajouter du contenu. Parmi celles-ci, nous avons utilisé Skin Composer, pour regrouper les textures utilisées par les boutons et les fenêtres de l'interface graphique. L'application regroupe les informations propres aux classes dans un fichier JSON, toutes les textures dans un fichier PNG et un fichier ATLAS pour permettre au framework de récupérer la texture voulue.

Pour importer les textures des cartes, nous avons utilisé GDX Texture Packer qui elle aussi ajoute les textures dans un fichier PNG, muni d'un fichier ATLAS. Cette application, nous a aussi aidé à corriger un bug d'affichage dû à LWJGL qui affichait des bordures noires sur les tuiles, en doublant les bordures.

Nous avons également utilisé le logiciel Tiled afin de dessiner les cartes facilement. En effet, le logiciel se comporte comme un logiciel de dessin, où l'on sélectionne la texture à déposer, et on peint les tuiles de la carte. Les différents plugins sont basés sur ce logiciel. En effet, le plugin d'importation a été pensé pour être compatible avec l'exportation d'une carte créée depuis Tiled.

Nous avons utilisé le logiciel SketchBook pour gérer la création des personnages dont les silhouettes proviennent de la base de offert par Matthew Krohn dans son dépôt GitHub - makrohn/Universal-LPC-spritesheet.

De la même, nous avons employé les images de @chickysprout https://twitter.com/chickysprout/status/966655304527048705 pour les arbres du biome volcano et les images de finalbossblues.com pour les palmiers du biome desert.

4.2.2. Choix artistiques:

Avec toutes les possibilités que nous avions devant nous, nous avons décidé de dessiner nos propres personnages, malgré l'investissement de temps en plus, pour donner un trait inclusif au projet. Ainsi pour les différents décors, nous avons pu rajouter toutes sortes de petits détails qui rendent l'atmosphère du jeu beaucoup plus agréable.

D'autre part, une bande sonore étant un élément clé de l'identité d'un jeu et ce groupe fournissant de nombreux musiciens et compositeurs nous avons naturellement commencé à composer nous même la bande originale du jeu. Les pièces, qui correspondent aux différents biomes ont été pensées pour un effectif musical correspondant aux talents des membres du groupe. Nous avions en effet au départ l'ambition d'interpreter nous même notre musique et de nous enregistrer en studio. Évidemment nous n'en avons pas eu le temps. Outre la contrainte de l'effectif, nous avons essayé d'écrire des musiques qui retransmettent au mieux l'ambiance que nous inspiraient les différents biomes. La composition s'est répartie comme suit : Menu : Vladislas, Grass : Soline, Ice : Vladislas, Volcano : Soline, Désert : Soline.

5 Documentation

4.1. Manuel de l'interface graphique

Pour lancer l'interface graphique, l'utilisateur devra entrer la ligne suivante en ligne de commande:

```
$  ./gradlew build
```

Puis:

```
$  ./gradlew run
```

4.2. Manuel de création de nouvelles cartes

Créer un dossier pour stocker les tuiles (sur les exemples Tiled, dont chemin absolu est C:\Tiled\), telecharger Tiled

```
$  ./gradlew select Plugin
```

Sur le scanner qui s'ouvre, lancer:

```
$  exportTexture --path=C:\Tiled\ --biome=grass
```

Ouvrir Tiled, créer un nouveau jeu de tuiles en sélectionnant le fichier C:\Tiled\grass.png, qui contient les textures précédemment exportées du jeu (paramètres: Largeur = Hauteur 64px, Marge = Espacement= 4px), puis créer une nouvelle carte (paramètres:L= H = 20 tuiles, L = H = 64px).

Une fois la carte dessinée il faudra l'exporter en fichier macarte.csv et l'enregistrer sur le dossier Tiled. Ensuite, pour l'ajouter à la base de donné du jeu. Il faudra, relancer la commande:

```
$  ./gradlew select Plugin
```

Puis :

```
$  importMap --paths=C:\Tiled\macarte.csv
```

De cette façon, l'utilisateur pourra jouer sur la carte que lui-même à créer.

Conclusion

Tout au long de ce projet nous avons été amenés à concevoir et implémenter un jeu de stratégie, basé sur le jeu Wargroove. Pour cela, le projet a été divisé en 2 groupes afin de mieux répartir les tâches et augmenter l'efficacité : Le groupe UI et le groupe Core. En particulier, les acquis du cours de programmation orientée objet étaient sans cesse sollicités et l'utilisation de libgdx nous a encore permis d'aller plus loin dans les possibilités de l'interface graphique et d'acquérir des nouvelles connaissances.

c'est ainsi que les objectifs obligatoires ont été réussis de plus que certains objectifs personnels que nous avons fixés et puis implémentés afin d'améliorer l'ambiance et l'expérience du jeu.

Le jeu peut encore évoluer. En effet, nous avons d'autres idées que nous voudrions implémenter dans l'avenir proche comme un serveur qui permettrait de faire des parties en mode multijoueur, une version pour android (apk) et encore des d'autres idées…

Annexe

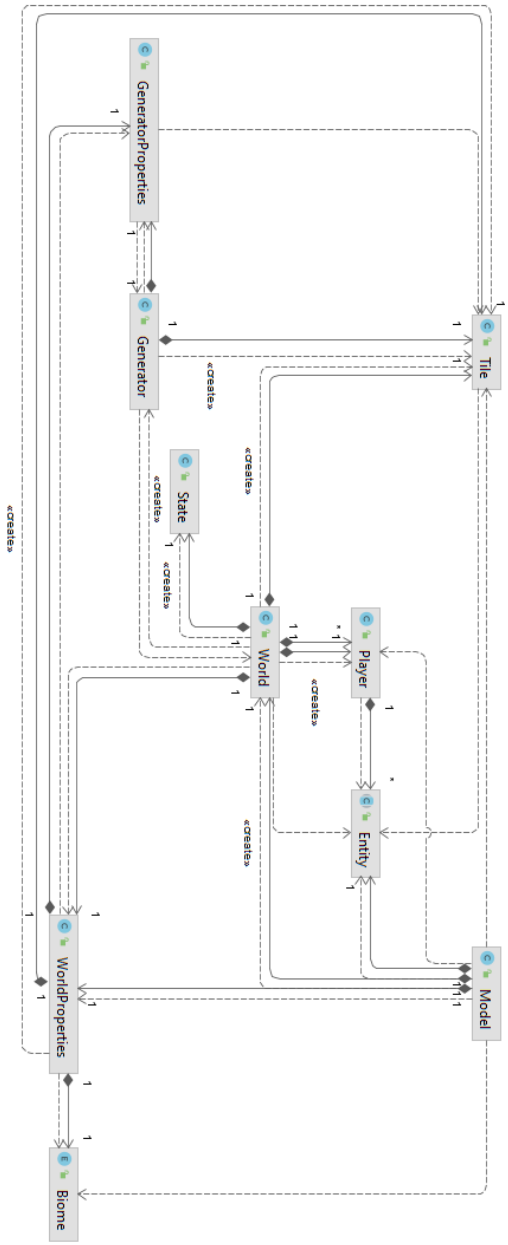


Figure 7 : Diagramme des classes illustrant le modèle

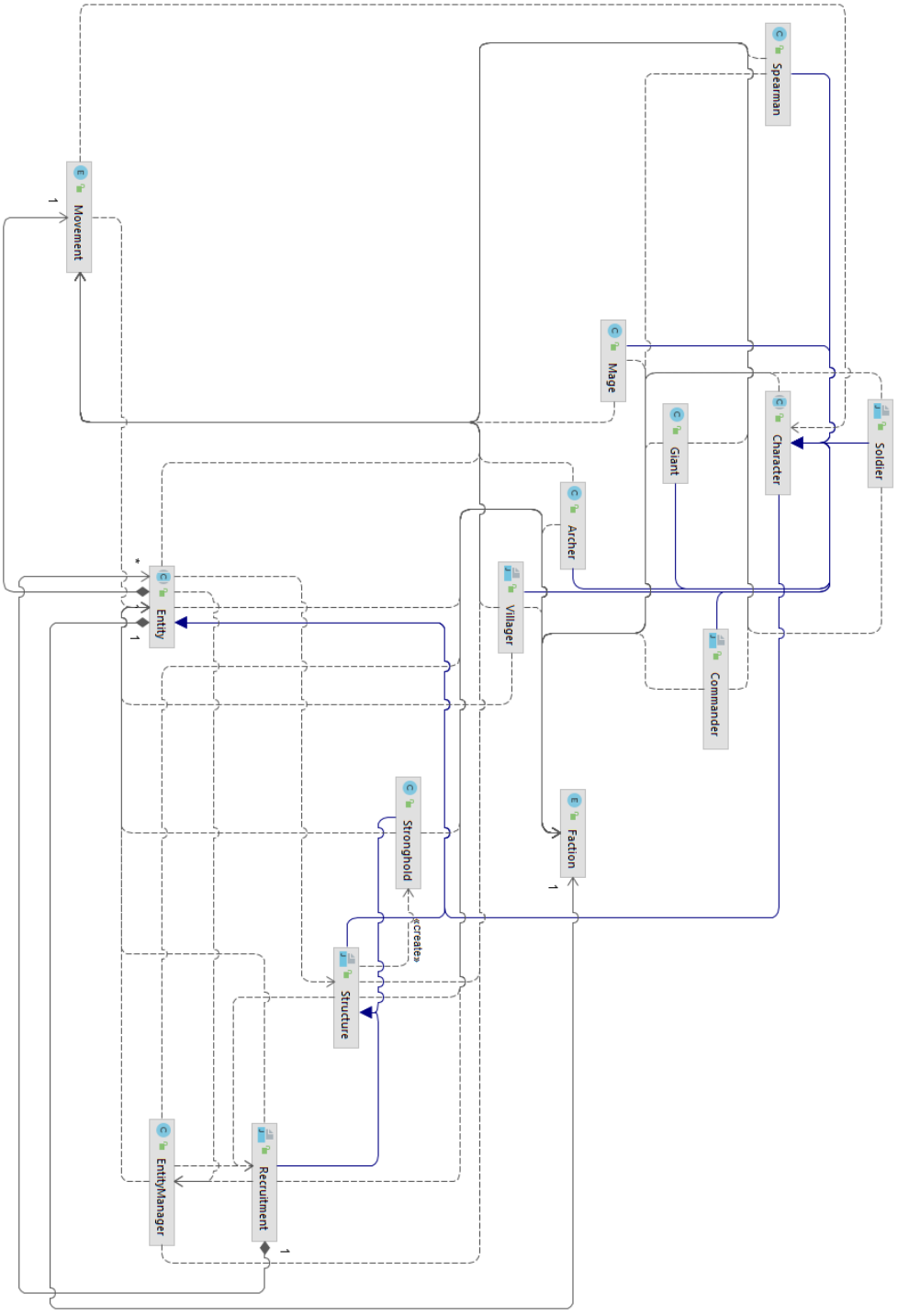


Figure 8: Diagramme des classes illustrants les entités

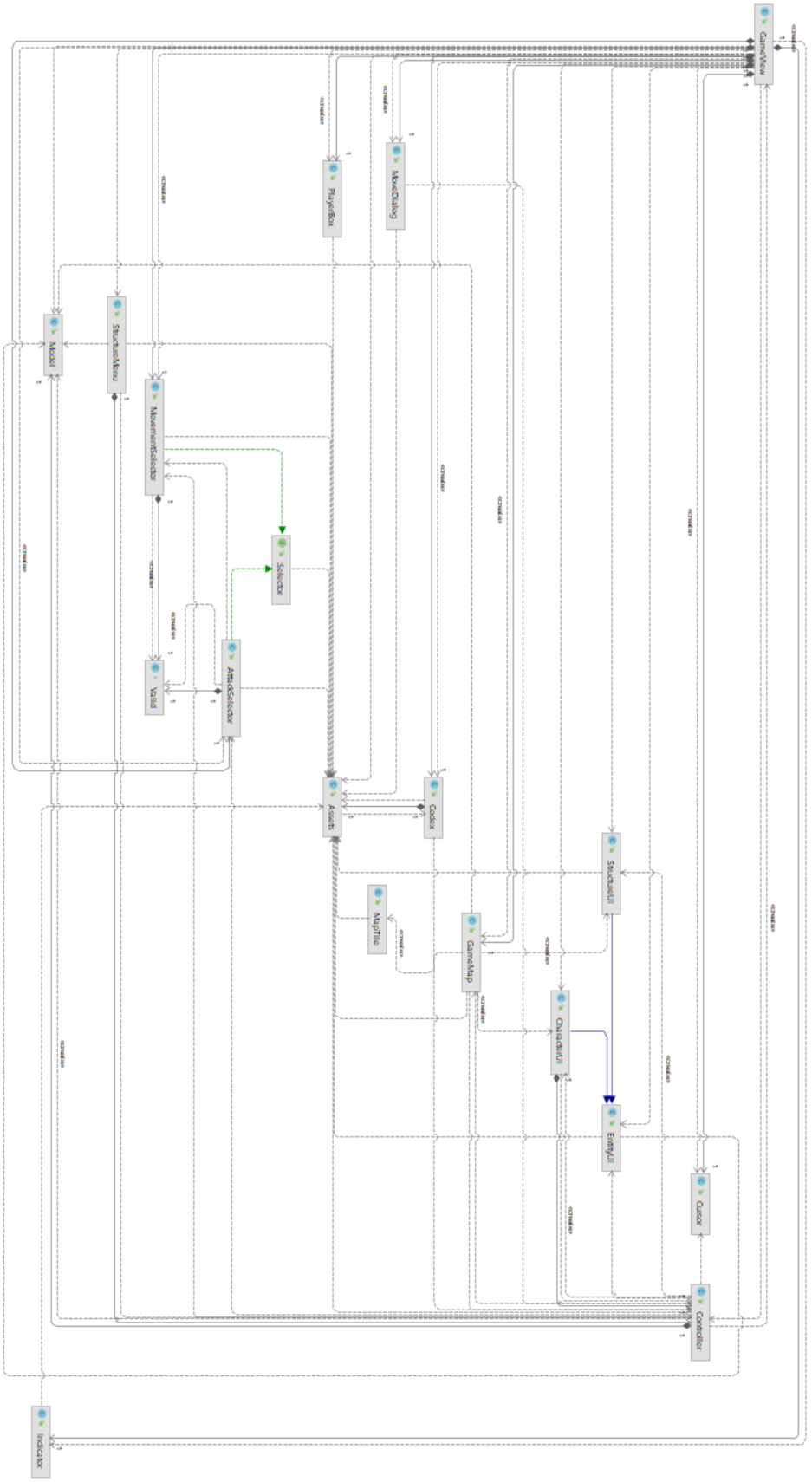


Figure 9: Diagramme des classes illustrant le gameView

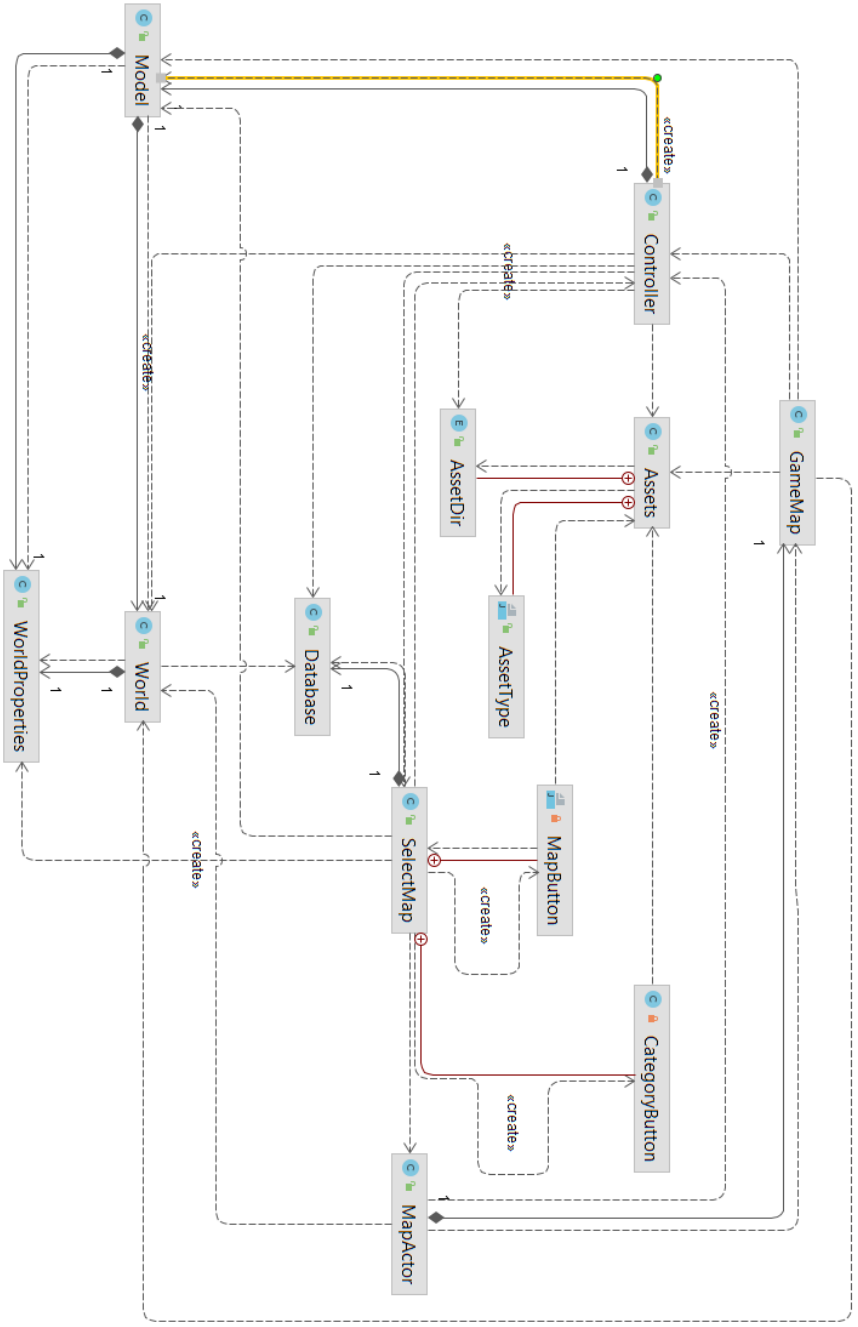


Figure 10: Diagramme des classes illustrant l'écran de sélection de la carte.

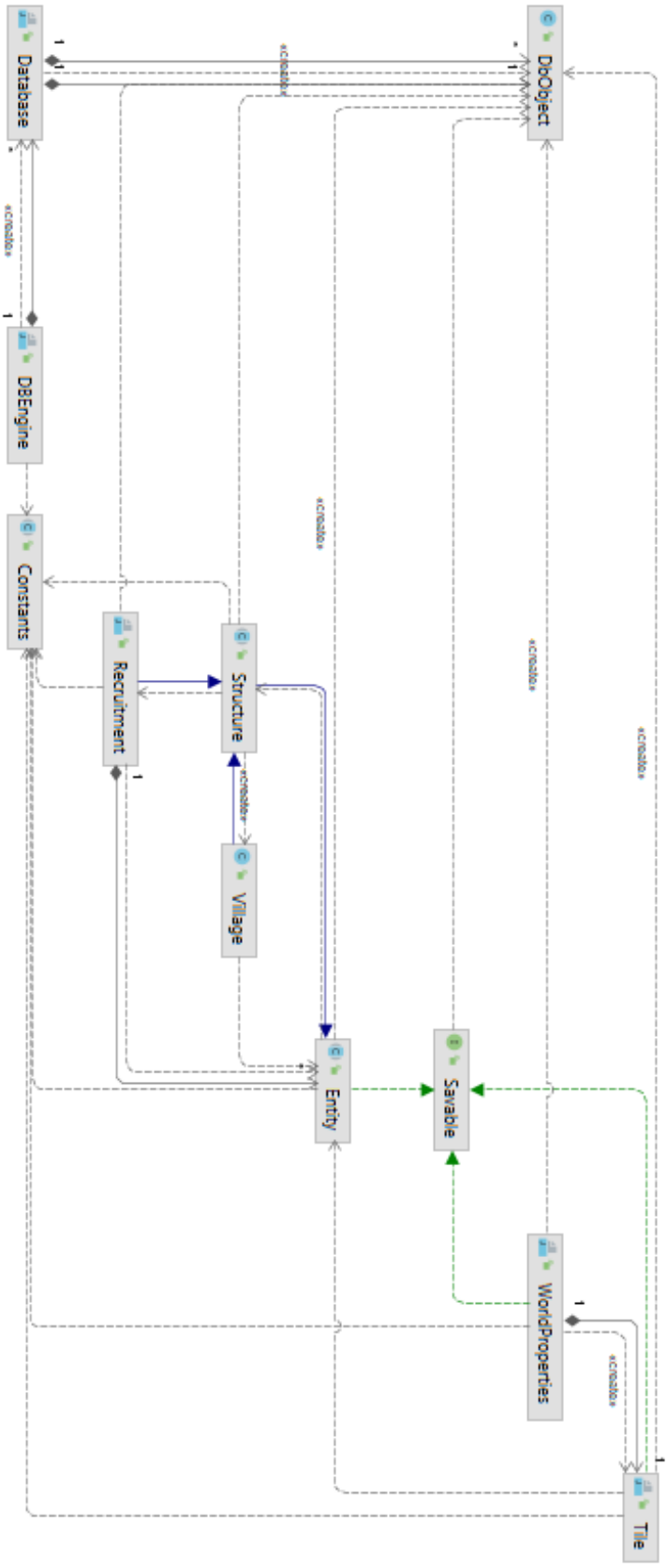


Figure 11: Diagramme des classes illustrant la sauvegarde

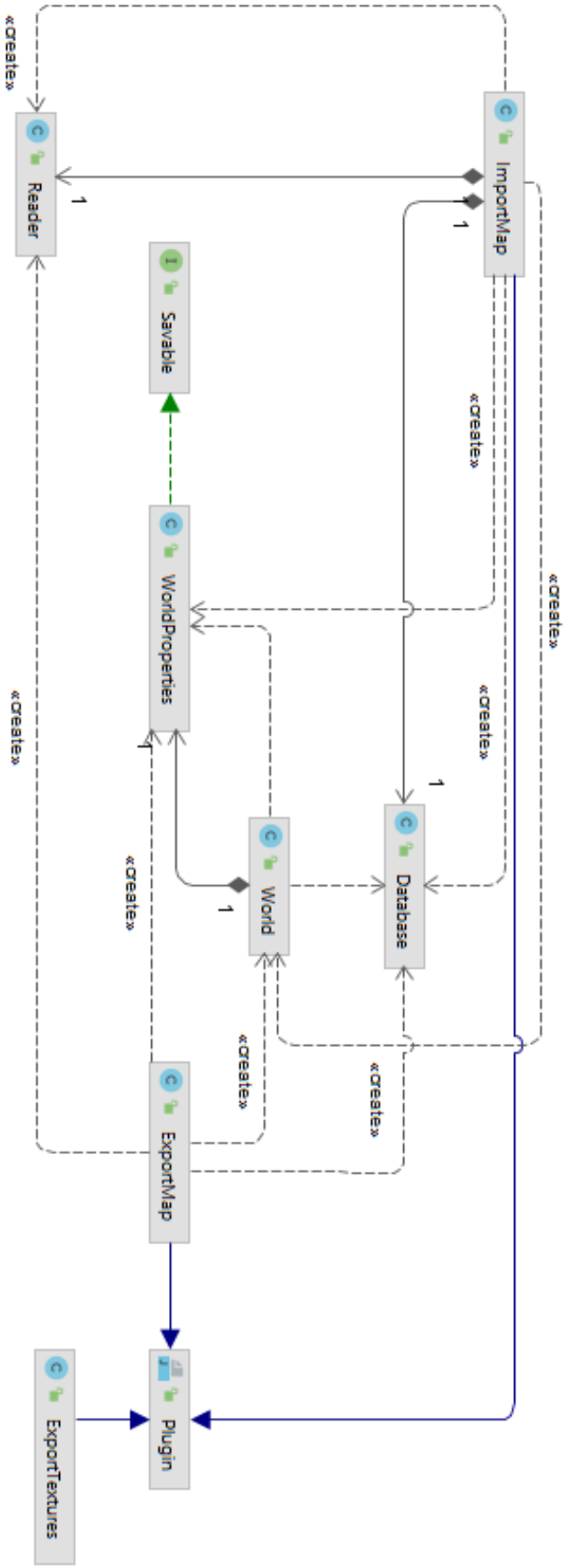


Figure 12: Diagramme des classes illustrant les plugins.