Projet Programmation Orientée Objet

Groupe 48

Illia Gargaun(Info 5), Lilian Legrand(Info jap)

(22007415) *(22007128)*

Les Colons de Catan

(https://github.com/SportDay/L2-Projet-POO3)

Sommaire

Installation

1. Vous devez avoir Java 10 ou bien une version plus récente.
2. Pour éditer et visualiser le code, on vous conseille d'utiliser une IDE.

Utilisation

Compiler le programme

Se mettre dans le dossier "src/", écrire une des deux commandes :

1. javac -cp main/java/ -sourcepath main/resources/ -d classes main/java/l2/poo3/*.java main/java/l2/poo3/*/*.java main/java/l2/poo3/*/*/*.java
2. javac -cp main/java/ -sourcepath main/resources/ -d classes main/java/l2/poo3/*.java main/java/l2/poo3/controller/gui/*.java main/java/l2/poo3/controller/terminal/*.java main/java/l2/poo3/model/*.java main/java/l2/poo3/model/Enum/*.java main/java/l2/poo3/model/PlayerType/*.java main/java/l2/poo3/model/CaseType/*.java main/java/l2/poo3/Other/*.java main/java/l2/poo3/view/*.java main/java/l2/poo3/view/gui/*.java main/java/l2/poo3/view/terminal/*.java

Pour obtenir un fichier .jar :

1. Se mettre dans le dossier "src/classes"
2. Copier le contenu du dossier "src/main/resources" dans le dossier "src/classes"
3. Ouvrir un terminal dans le dossier "src/classes"
4. Écrire la commande : jar cvfm LesColonsDeCataneGroupe48.jar META-INF/MANIFEST.MF *

Lancer le programme

Pour lancer le programme vous avez plusieurs possibilités :

1. En doublant cliquant sur le fichier LesColonsDeCataneGroupe48.jar
2. En ouvrant un terminal dans le dossier contenant le fichier LesColonsDeCataneGroupe48.jar et en exécutant la commande java -jar LesColonsDeCataneGroupe48.jar
3. Si vous avez compilé vous-même le projet :
   3.1. Exécuté cette commande dans le terminal : java -cp classes/ l2.poo3.CatanMain

3.2. Si vous avez une erreur de type :

```
Exception in thread "main" java.lang.NullPointerException
        at java.desktop/javax.swing.ImageIcon.<init>(ImageIcon.java:217)
        at l2.poo3.CatanMain.main(CatanMain.java:18)
```

Alors il faut que vous copier touts les fichiers et dossier qui se trouve dans le dossier src\main\resources vers le dossier src\classes, vous devez obtenir le contenu suivant dans le dossier classes.*(voir image 0)*

Dans tous les cas le programme vous demandera le type d'affichage, vous avez le choix entre l'affichage textuel et l'affichage graphique.*(voir image 1)*

Vous pouvez directement lancer un de ces deux affichages en exécutants les commandes suivantes :

Pour l'affichage graphique :

1. java -jar LesColonsDeCataneGroupe48.jar gui
2. java -jar LesColonsDeCataneGroupe48.jar graph
3. java -cp classes/ l2.poo3.CatanMain gui
4. java -cp classes/ l2.poo3.CatanMain graph

Pour l'affichage textuel :

1. java -jar LesColonsDeCataneGroupe48.jar text
2. java -jar LesColonsDeCataneGroupe48.jar terminal
3. java -cp classes/ l2.poo3.CatanMain text
4. java -cp classes/ l2.poo3.CatanMain terminal

Présentation Du Jeu

Ce projet est basé sur le jeu de société nommé Les Colons de Catane, qui a été créé en 1995. Chaque joueur contrôle plusieurs colonies et villes, au début de chaque tour les joueurs lancent deux dés, le résultat de ces deux dés permet de générer des ressources, ces ressources permettent de construire des nouvelles colonies, villes, routes ou bien d'acheter des cartes de développement qui ajoutent des fonctionnalités très intéressantes. En faisant ceci les joueurs accumulent des points dits points de victoire, le premier qui a atteint dix points de victoire à gagner, cependant si un autre joueur possède des cartes de développement de type point de victoire (qui reste caché jusqu'à la fin du jeu) il pourrait gagner.

Présentation Du Projet

Tout notre projet est basé sur une architecture de type MVC pour Modèle-vue-contrôleur. Nous avons implémenté les fonctionnalités importantes de ce jeu de société, en utilisant le langage Java (version 10), et en utilisant les API inclus dans le SDK 10, c'est-à-dire que nous n'avons pas utilisé des bibliothèques externes (écrit par d'autres personnes)

Pour la version textuelle on décide d'afficher tout le jeu dans un terminal, ce qui permet de lancer et de jouer à ce jeu même dans un environnement sans interface graphique (ex : sur des serveurs).

Pour la version graphique, on utilise l'API Swing et AWT fourni de base dans l'API java.

On implémente le cahier de charge minimale qui nous a été fourni, et on a ajouté des fonctionnalités supplémentaires qui vont être détaillées dans la suite de ce rapport.

Cahier de Charge Minimale

Paramétrage Du Jeu

Nous avons implémenté la configuration de base demander dans le cahier de charge, c'est-à-dire le choix du nombre de joueurs entre 3 et 4, le choix de décider si un joueur est “humain” ou bien c’est une “IA”, en plus on décide d'autoriser aux utilisateurs de choisir la taille du plateau, pour la version textuel la taille maximale du plateau peut être de 11 cases (avec des ressources) en longueur et de 3446 cases en largeur, pour la version graphique la taille maximale peut être à infini (c'est-à-dire n’importe quels nombres choisis par le joueur) cependant pour des choix de performance, on a limité la longueur et la largeur à 9999 cases, la taille minimale du plateau pour les deux versions est de 4x4 cases. L’utilisateur peut aussi choisir le nombre de points de victoire maximale.

La Création d’un plateau

La création du plateau est aléatoire, c'est-à-dire le nombre de cases de chaque type, leur chiffre ou bien le type de port est aléatoire, et donc c’est très peu probable qu'on obtient un tableau, sur lequel on a déjà joué.

On a eu beaucoup de problème pour l’affichage du plateau dans la terminale, car on ne savait pas comment l’afficher, on avait plusieurs idées comme un affichage simple avec des commandes pour afficher les détaille d’une case, mais on a décidé de faire un affichage plus détaillé pour éviter aux joueurs de mémoriser des commandes supplémentaires.

Pour l’affichage du plateau dans l’interface, on n'a pas eu de problèmes particuliers, car on a affiché le même type de plateau.

Pour voir les images des plateaux : plateaux terminal *(voir image 2)*, plateaux graphiques *(voir image 3)*.

Implémentation Des Règles Du Jeu

Construction des routes, villes, cités :

- Partie textuelle :
    - La construction se fait grâce aux commandes (cb) pour construire un bâtiment [(col) pour une colonie, (vil) pour une ville], (br) pour construire une route.
- Partie graphique :
    - La construction se fait grâce à un panneau ou l’utilisateur peut choisir ce qu'il veut construire *(voir image 4)*
    - Après avoir choisi un type de construction sur le plateau toutes les cases où il peut construire deviennent vertes (le contour). *(voir image 5)*

Consulter ces ressources :

- Partie textuelle :
    - En exécutant la commande (cr) le nombre de ressources du joueur courant est affiché *(voir image 6)*

- Partie graphique :
    - Les ressources des joueurs sont affichées en bas à droite de la fenêtre. *(voir image 7)*

Personnage du voleur (cas de 7 aux dés) :

C'est le même principe que pour la construction de bâtiments sauf que dans ce cas-là, on déplace le voleur sur une case qui produit une ressource (ou le désert), si à côté de cette case un joueur possède une ville ou colonie le joueur qui a obtenu 7 lui vole une ressource choisit au hasard, si les joueurs ont plus de 7 ressources au totale ils doivent les supprimer, il les choisit lui-même les ressource à supprimer.

- Dans la partie textuelle cela se fait à l'aide de commandes, l'utilisateur ne pourra pas quitter cette partie sans supprimer les ressources nécessaires.
- Dans la partie graphique cela se fait à l'aide d'un menu qui s'ouvre automatiquement et qu'il est impossible de fermer sans supprimer les ressources.

Cartes spéciales de développement :

Cette partie, nous à poser quelques problèmes tel que quels types de carte ajouter et le plus grand problème comment les implémenter. Comme on n'a pas eu assez de temps, on a décidé d'implémenter les 5 types de cartes présents par défaut dans le jeu original :

- Chevalier
    - On a utilisé le même principe que pour le 7 aux dés.
- Monopole
    - Le joueur choisit une ressource à obtenir
    - Tous les autres joueurs doivent donner leur stock de cette ressource
- Invention
    - Le joueur peut choisir 2 ressources à obtenir
- Route
    - Le joueur obtient les ressources pour construire 2 routes
- Point De Victoire
    - C'est une carte qui donne un point de victoire au joueur qui l'obtient
    - Mais ce point n'est pas rajouté en nombre de points visible par les autres joueurs.
    - Ce point de victoire est ajouté à la somme totale de point de victoire, ce qui permet de gagner une partie même si l'on n'a pas atteint 10 point de victoire (par défaut)

Les Ports :

L'utilisation des ports ne nous a pas posé de problèmes, car on réutilise les mécaniques qu'on a implémentées dans d'autres fonctions.

- Pour la partie textuelle
    - On utilise toujours des commandes.
- Pour la partie graphique
    - On réutilise quelques interfaces qu'on utilise dans les autres fonctions

La plus grande armée :

Cette partie était la plus facile car on utilise simplement un compteur de chevalier pour chaque joueur, ce compteur est mis à jour à chaque fois que des joueurs jouent une carte de chevalier.

La route la plus longue :

Cette partie nous a pris beaucoup de temps, car on ne savait vraiment pas comment compter ce nombre, car une intersection pouvait avoir beaucoup plus de route qu'une autre intersection. Après on s'est rendu compte que c'est un peu le même problème que pour compter la hauteur d'un arbre binaire. Donc on a appliqué cet algorithme un peu modifié dans notre projet, en quelques mots, l'algorithme est divisé en plusieurs algorithmes, la fonction *findLastRoad* *(ligne 417 PlateauxModel.java)* renvoie les coordonnées de la route qui se trouve la plus loin dans le chemin, puis ces coordonnées sont utilisées dans la fonction *checkRoad* *(ligne 484 PlateauxModel.java)* qui se charge de parcourir le chemin dans l'autre sens pour cette fois calculer la longueur totale de la route, ces deux fonctions sont récursives et sont gérées par une autre fonction *getLengthBiggestRoad* *(ligne 401 PlateauxModel.java)* qui les appelle dans le bon ordre et fait passer les informations de l'une à l'autre. Les fonctions *findLastRoad* et *checkRoad* utilisent une autre fonction *endRoad* *(ligne 550 PlateauxModel.java)* qui indique si c'est la fin de la route ou non. Le problème, c'est que l'algorithme ne marche pas très bien il peut afficher une ou deux routes en plus, mais cela reste très rare

"L'intelligence Artificielle" :

"L'intelligence Artificielle" n'a rien d'un intelligent, car on génère des nombres au hasard qui vont décider où sur le plateau et quels types de bâtiment elle va construire (route, colonie, ville), ou bien quelles ressources elle doit supprimer (si elle plus de 7 ressources et un 7 au dé a était lancer).

En clair quand c'est le tour de "IA" elle vérifie :

1. S'il faut qu'elle supprime des ressources.
    1.1. Si oui elle les supprime
2. Si elle peut lancer les dés (sachant que pendant les 2 premiers tours personne ne peut lancer les dés)
    2.1. Si oui elle les lance
3. Si elle peut construire une ville
    3.1. Si oui elle le fait
4. Si elle peut construire une colonie
    4.1. Si oui elle le fait
5. Si elle peut construire une route
    5.1. Si oui elle le fait

Pour être un peu en égalité avec un joueur "humain", avec 33% "l'IA" ne pourra pas construire de plus une colonie / ville que le joueur "humain" qui en a le moins.

Arborescence

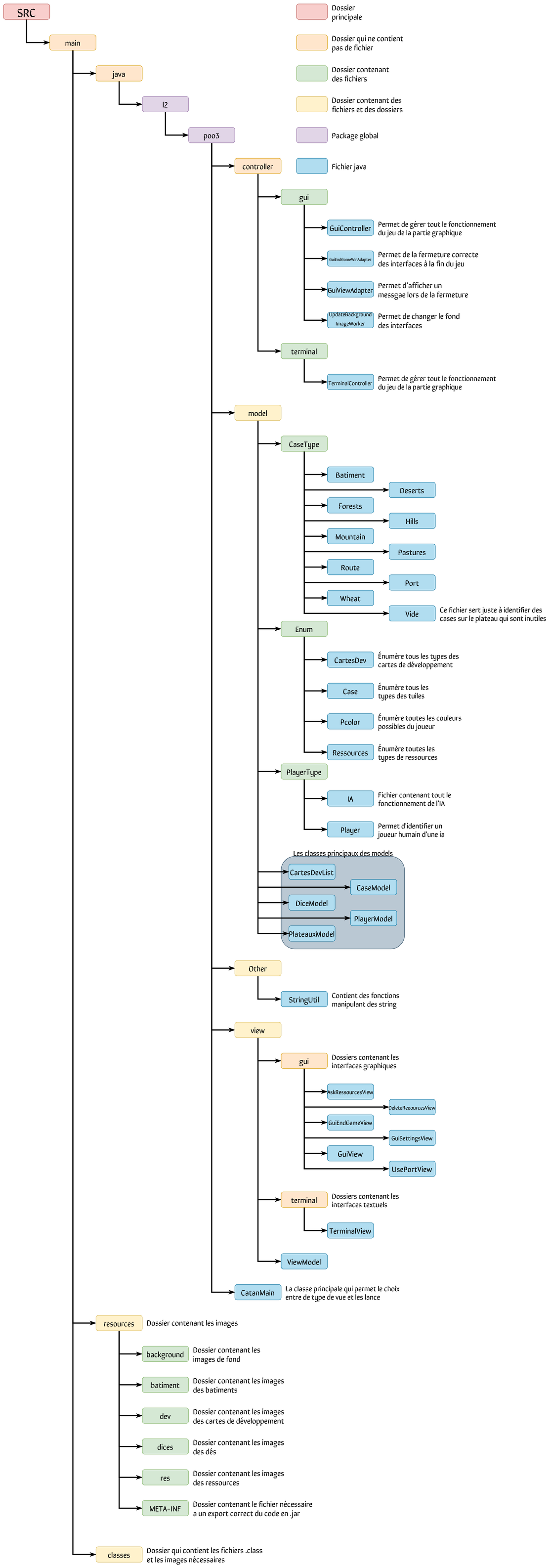
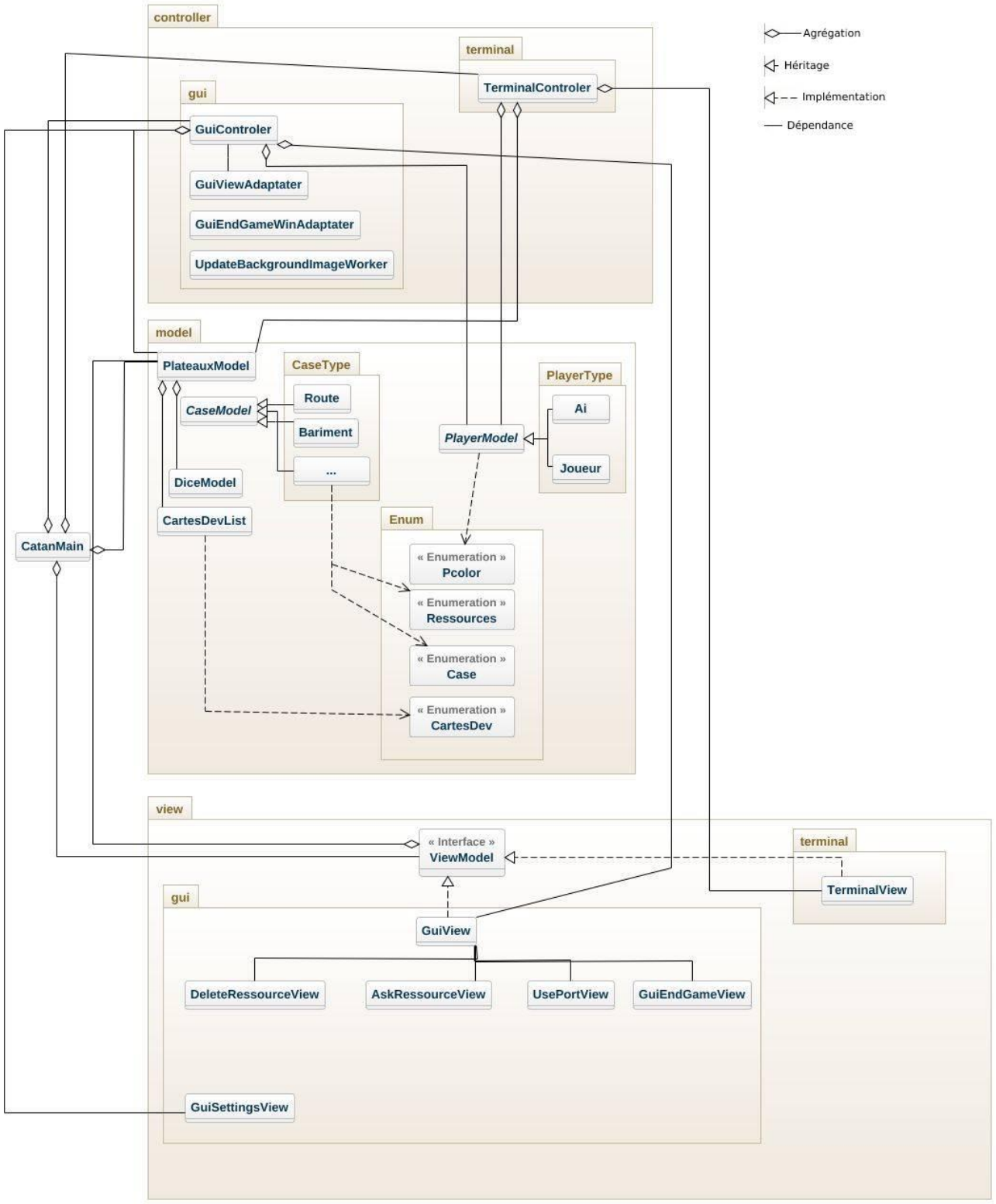


Diagramme de classe



Images

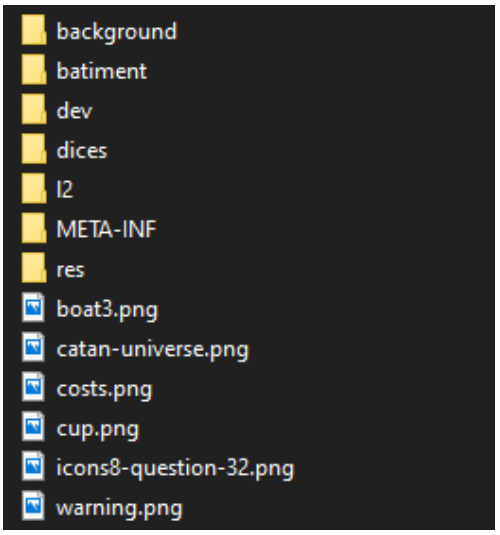


Image 0

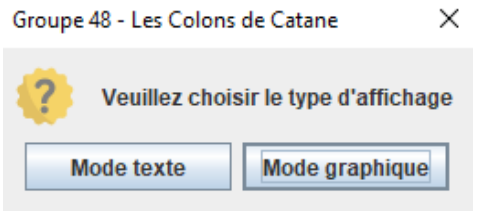


Image 1

y\x	1	2	3	4	5	6	7	8	9	10	11	12	13
A			Port 3:1				Port 3:1				Port Mouton 2:1		
B													
C			Foret 8		Colline 5		Montagne 9		Champs 4		Pre 10		
D													
E	Port 3:1		Champs 8		Champs 9		Colline 2		Colline 9		Montagne 3		Port Mouton 2:1
F													
G			Foret 12		Foret 5		Deserts 2 Voleur		Foret 10		Pre 12		
H													
I	Port Mouton 2:1		Champs 9		Foret 9		Montagne 3		Pre 11		Colline 8		Port 3:1
J													
K			Pre 3		Foret 5		Foret 9		Champs 2		Foret 4		
L													
M			Port 3:1				Port 3:1				Port 3:1		

Image 2

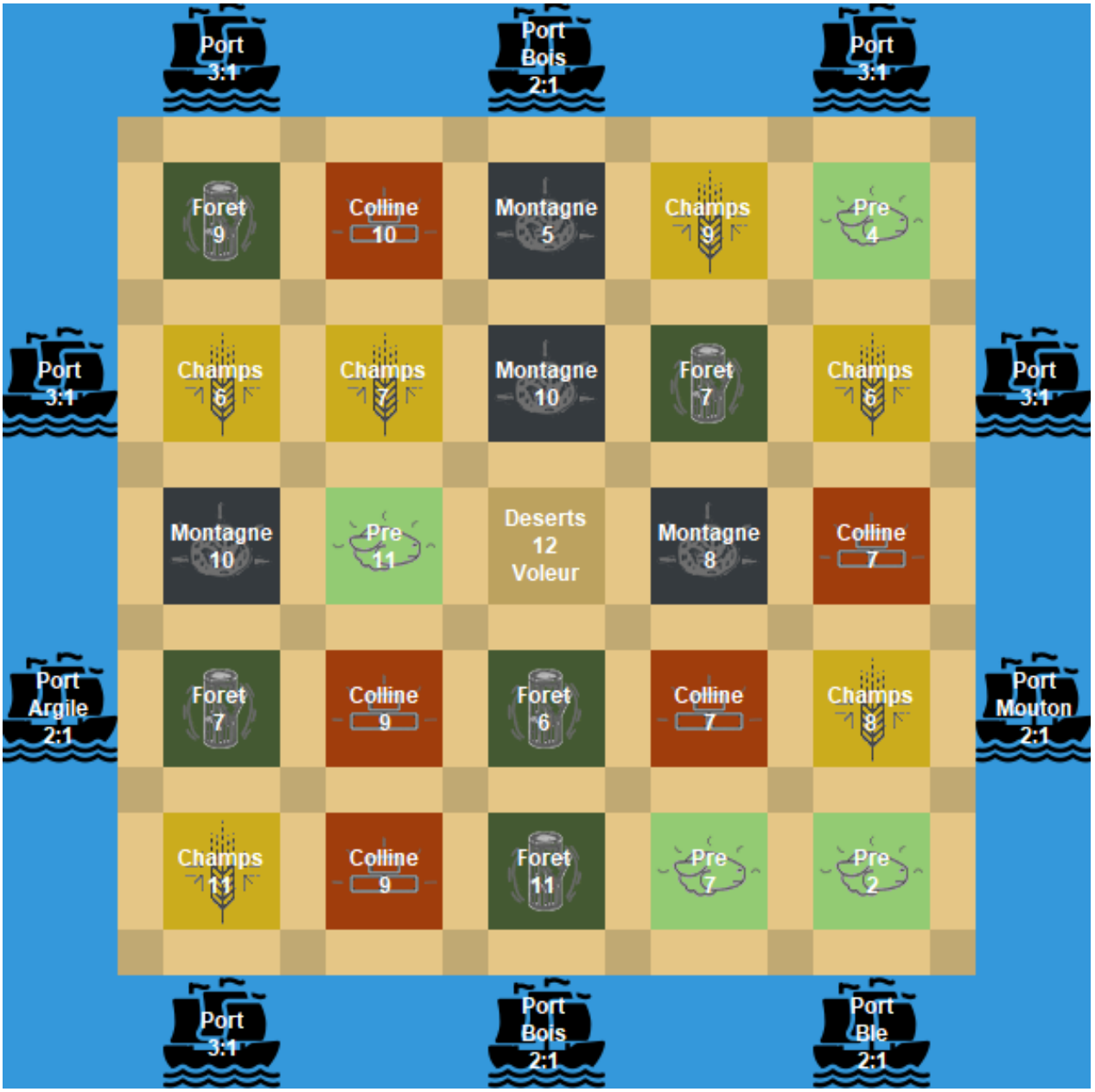


Image 3

Image 4

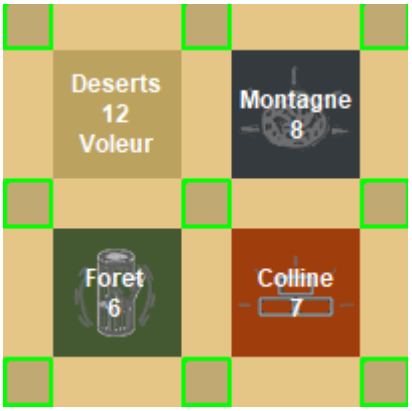
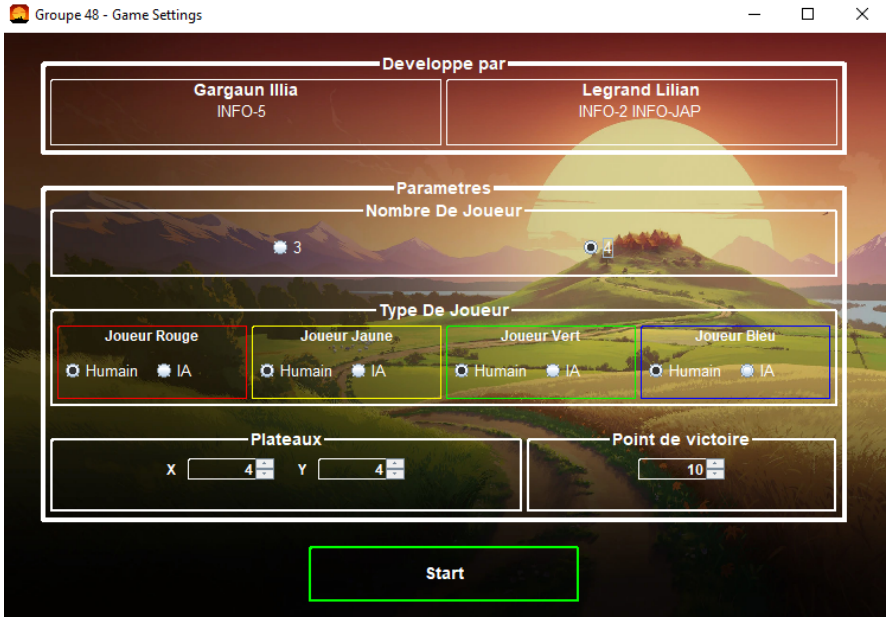


Image 5

Ressources du Joueur: Rouge					
Ressources	Bois	Ble	Argile	Minerai	Mouton
Quantite	3	1	3	0	1

Image 6

Image 7



Fenêtre de paramétrage du jeu partie graphique

L'interface du jeu de la partie graphique