Projet AP4B : Sim Power

Rapport final projet WejCity

CLOAREC Florian, HUMBERT Célian, AZANCOTH-BOSCAROL Carlo, BAUDOT Mathéo

Table des matières

1. Introduction

Le but de ce projet est de concevoir et de développer un jeu de gestion dans lequel le joueur devra créer une ville. Pour ce faire, il devra gérer la création des différents bâtiment produisant ou consommant de l'énergie tout en prêtant attention à ses ressources, à la pollution, à la satisfaction des habitants… Le langage utilisé pour ce projet est Java. Mais dans ce rapport, vous trouverez les différents supports nous aidant pour le développement de ce jeu, notamment des diagrammes. Tels que le diagramme de classe, le diagramme de cas d'utilisation, quelques diagrammes de séquence et des Scenarii au format texte. Ces diagrammes sont susceptibles d'évoluer au cœur du développement, mais, permettent effectivement de conceptualiser le projet de façon plus précise et d'anticiper des éventuels problèmes de développement et ainsi la réalisation. Vous trouverez également une explication des diagrammes ainsi qu'une explication des principaux choix d'implémentation. (l'ensemble des diagrammes présent dans ce rapport sont fourni en annexe pour une meilleure lisibilité)

2. Phase de conception

a. Diagramme de classe

Ce diagramme de classe représente toutes les classes et les liens entre les classes auxquelles nous avons pensé. Ce diagramme permet de se rendre compte de la manière dont on accédera aux différents attribut et méthodes.

Afin de stocker tous les éléments du jeu, on a créé la **classe Map** qui contient un tableau à double entrée de Plot ainsi qu'un objet de type Inventory. Ainsi lors du lancement d'une partie un objet de type Map est créé et on stocke ainsi en mémoire toutes les informations de la partie.

La classe Plot permet de représenter une case de la Map, elle est composée d'un attribut Construction qui vaut soit Null si rien n'est construit sur cette case soit la référence de la construction de la case. Elle possède aussi un attribut booléen qui indique s'il est possible de construire sur la case. Ainsi qu'un attribut de type Resource qui représente les ressources minières que l'on peut trouver dans le sous-sol de cette case. De plus, il y a un attribut qui indique quel est le type de la case (terre, herbe, sable, eau). On définit notamment une méthode qui permet de générer de façon aléatoire un nouvel objet de type Plot afin de générer la carte de jeu en début de partie.

La classe Inventory permet de représenter toutes les ressources que le joueur possède dans la partie. Un objet de cette classe est composé d'une liste de Resource.

Un objet de type Resource est composé d'un attribut de type ResourceType qui permet d'indiquer la nature de la ressource et d'un attribut de type int qui indique la quantité de cette ressource. **La classe Resource** est surtout utilisée dans l'inventaire, mais aussi dans les constructions pour connaître le coût de construction et dans les bâtiments pour indiquer les entrées et les sorties.

Tous les objets qui peuvent être construits sur une case de la carte de jeu héritent de la **classe abstraite Construction**. Cette classe à comme attribut la position de type Point, le coût de construction de type Resource et la nature de la construction de type ConstctionType. Cette classe définit aussi les fonctions abstraite build(Point), update() et destroy() qui devront être implémentées que par les classes filles.

La classe Point nous permet de représenter un point, qui est simplement composé d'une coordonnée en x et en y.

Les classes Road et Tree héritent toutes les deux de la classe Construction et représentent respectivement une route et un arbre. Les arbres sont placés aléatoirement sur certaines cases lors de la génération de la carte. Les routes le sont aussi, à l'exception qu'elles sont toutes adjacentes.

La classe abstraite Building hérite de la classe Construction, elle à comme attribut un Pylon qui est le pylône lié à ce bâtiment. Et un BuildingParameter qui contient tous les paramètres du bâtiment.

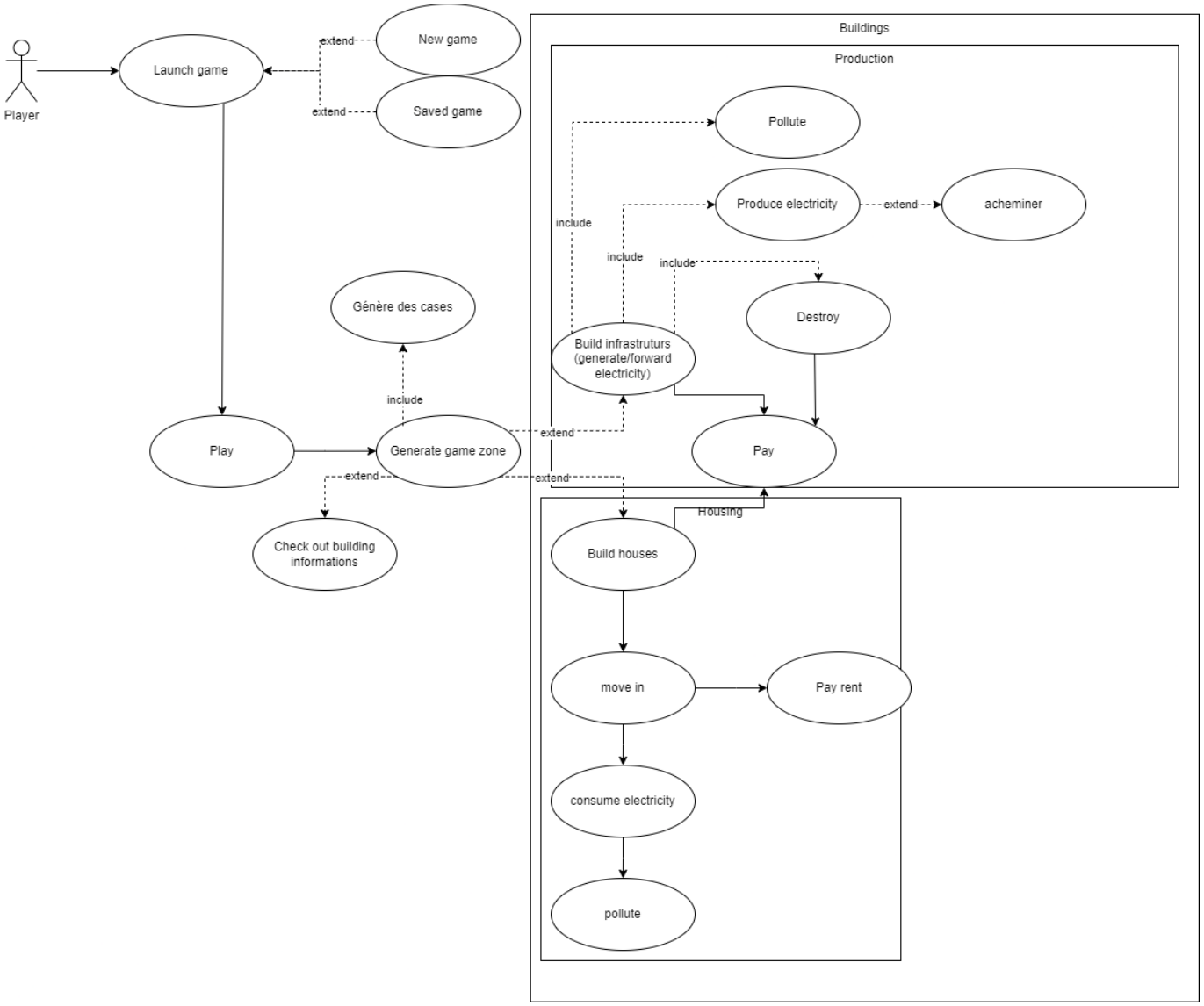
Les classes CoalPlant, Driller, EntertainementPark, Farm, House, SolarPanel, NuclearPlant, WindMill héritent toutes de Building et permettent de construire les différents bâtiments du jeu.

La classe Pylon hérite de Construction elle contient en comme attribut une liste de Building qui représente les bâtiments voisins et une liste de Pylon qui correspond au pylône relié à ce pylône.

La classe ElectricalNetwork représente le réseau électrique et contient en attributs une liste Pylon ainsi qu'une liste de Building.

La classe FileUsage est la classe qui sert à lire et écrire des fichiers.

b. Diagramme de cas d'utilisation



Le diagramme des cas d'utilisation nous permet de voir quelles sont les actions que l'utilisateur peut effectuer. Dans ce cas, nous pouvons voir l'utilisation d'un menu qui s'ouvre au lancement du jeu. Ce menu nous permet de pouvoir lancer une nouvelle partie où de lancer une partie préalablement sauvegardée. Une fois la partie lancée on peut s'intéresser à toutes les actions que l'utilisateur peut faire en jeu.

La première chose qui va être faite et qui ne dépend pas de l'utilisateur est la génération du terrain qui va donner une caractéristique de pollution et de ressources. Mais également la constructibilité, la production, la consommation énergétique et si la case contient un bâtiment où non. Toutes ces informations seront visibles par l'utilisateur afin d'optimiser au mieux la gestion de son terrain.

La seconde fonctionnalité que pourra utiliser le joueur sera la création des habitats qui vont être placés sur les cases sélectionnées, ces bâtiments pourront ainsi créer de la richesse et de la population en échange d'une certaine pollution et consommation d'énergie. De plus, plusieurs actions seront possibles sur les bâtiments, comme les supprimer, les déplacer ce qui pourra permettre à l'utilisateur de restructurer sa carte en fonction des différentes contraintes qu'il rencontrera dans l'optimisation de sa ville.

La deuxième grosse fonctionnalité est la possibilité de placer des bâtiments industriels, premièrement les bâtiments de production énergétique comme les centrales électriques et les lignes électriques qui permettent l'alimentation en électricité des différents bâtiments. Les bâtiments de production de ressources, l'utilisateur pourra construire des bâtiments qui vont exploiter les ressources de la case dans laquelle il se situe. Ces ressources permettront de construire de nouvelles infrastructures.

Pour finir des routes devront être placées pour permettre la circulation dans la ville, puisqu'un bâtiment devrait se trouver obligatoirement au bord d'une route. On pourra également ajouter un système de vues pour que l'utilisateur se rende compte plus facilement de la pollution, de la richesse ou alors de la production de certaines zones. Mais pour le moment cela reste un objectif secondaire si le projet avance bien.

c. Scenarii

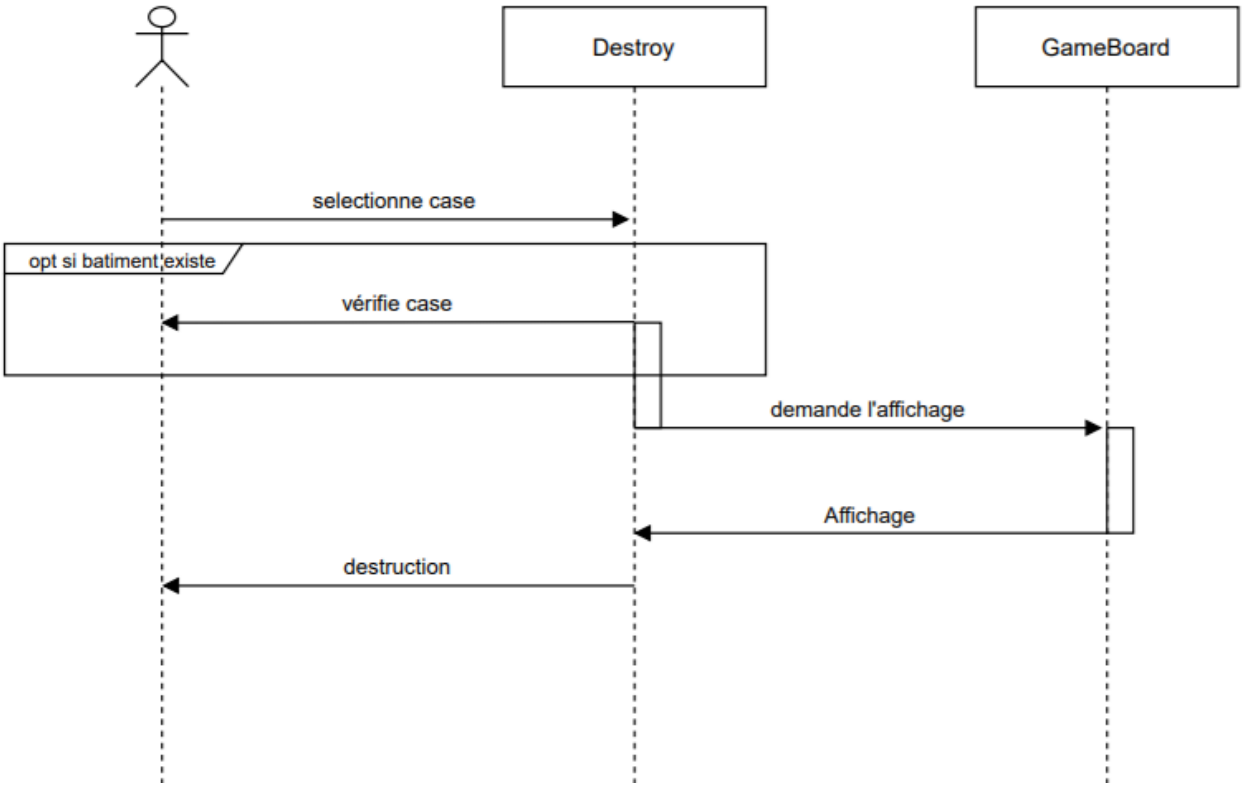
voir : Scenarii.pdf

Le but des scenarii est de décrire textuellement en détail le déroulement en étapes des différentes utilisations possibles. On part d'un scénario principal et on peut éventuellement en montrer les variantes en créant des branches. Ce sont les scénarios alternatifs ou encore les scénarios d'erreurs. Une fois leur résolution, on repart sur le chemin du scénario principal à une étape donnée.

Nous avons décrit certains des scenarii les plus évidents qui nous semblaient essentiels, tels que la construction, ou la destruction d'un bâtiment, ou encore le lancement du jeu. Mais aussi des scenarii moins importants sur des points tels que la production ou la consommation d'électricité, même si ses points paraissent manifeste les scenarii permettent d'anticiper les déroulements alternatifs.

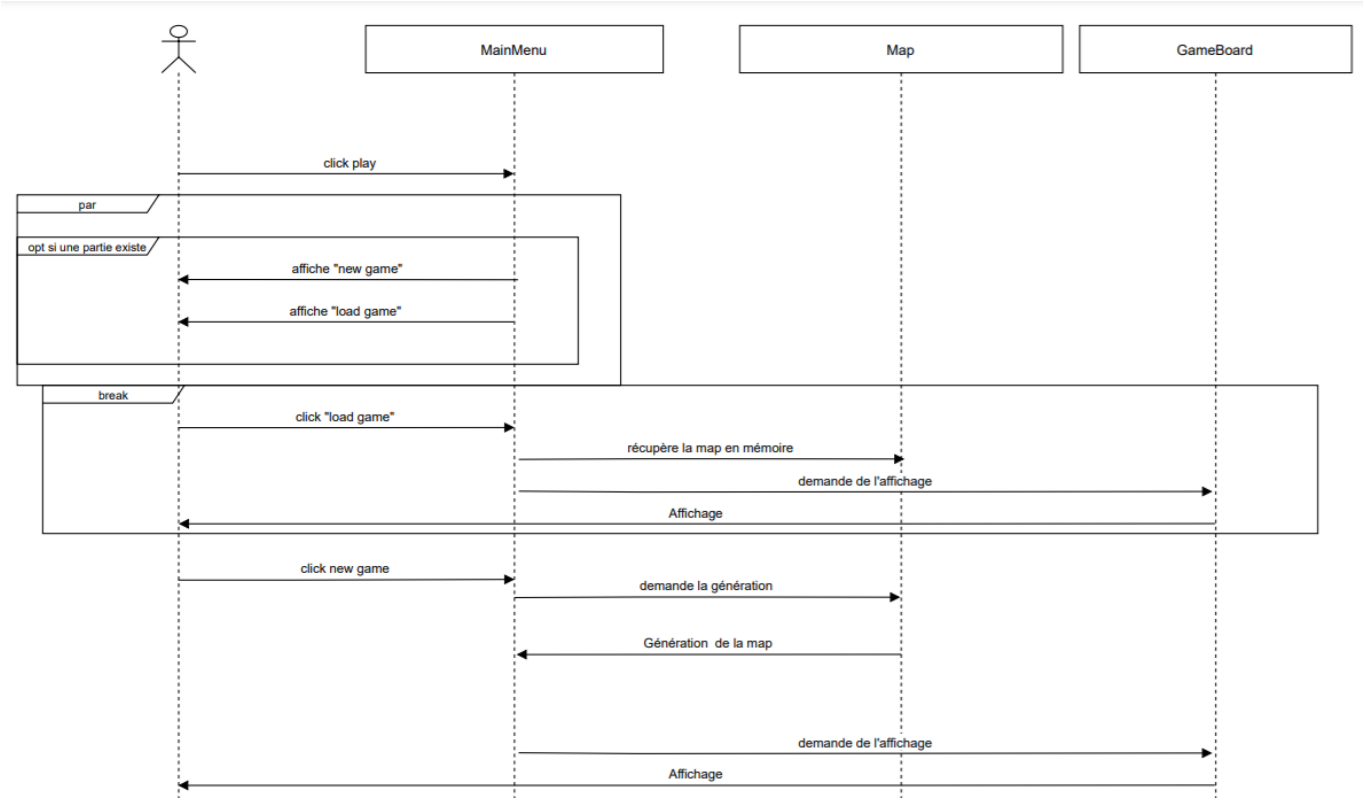
d. Diagrammes de séquence

i. Destroy



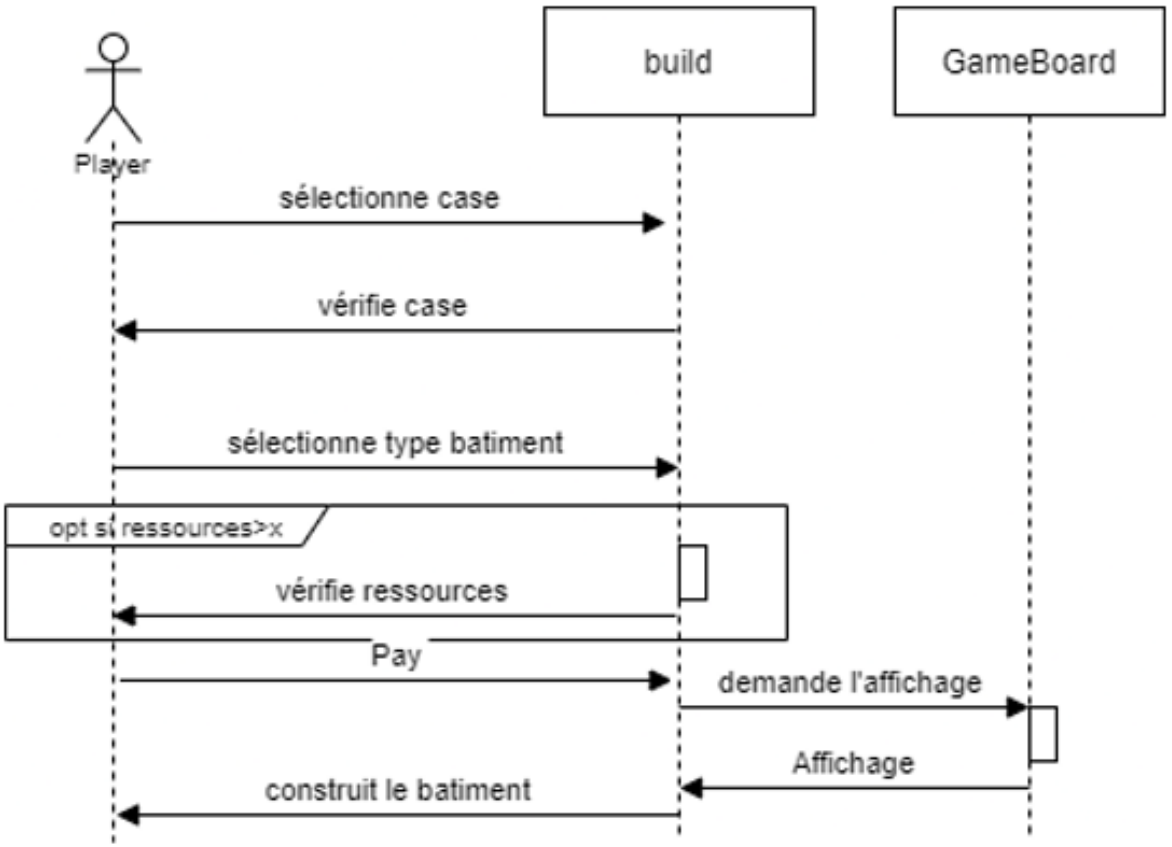
Ce diagramme de séquence représente la destruction d'un bâtiment, il s'agit d'une séquence relativement simple, on voit cependant qu'il faut prêter attention à la vérification de l'existence du bâtiment à détruire pour pouvoir le détruire, on fait également apparaître la classe "GameBoard" qui est la classe qui nous permettra de gérer l'affichage des différents éléments présent sur le plateau de jeu.

ii. Play



Ce diagramme de séquence représente ce qu'il devrait se passer lorsque le joueur interagit avec le menu principal. Le joueur aura le choix de commencer une nouvelle partie ou reprendre une partie existante si elle existe, si c'est le cas la classe map devra récupérer les informations enregistrées et le transmettre à la classe GameBoard afin de l'afficher, dans l'autre cas, la classe map devrait générer la map et transmettre les informations nécessaires à la classe Gameboard pour pouvoir afficher. La classe MainMenu ne sert que d'intermédiaire entre le joueur et le reste des classes pour le lancement du jeu.

iii. Build



Ce diagramme de séquence montre la séquence de construction d'un bâtiment, on remarque que les actions à surveiller sont l'état de la case on le joueur veut construire (vérifier qu'elle soit vide et constructible) et les ressources du joueur (ne pas permettre au joueur de construire s'il n'a pas les ressources nécessaires).

e. Gameplay

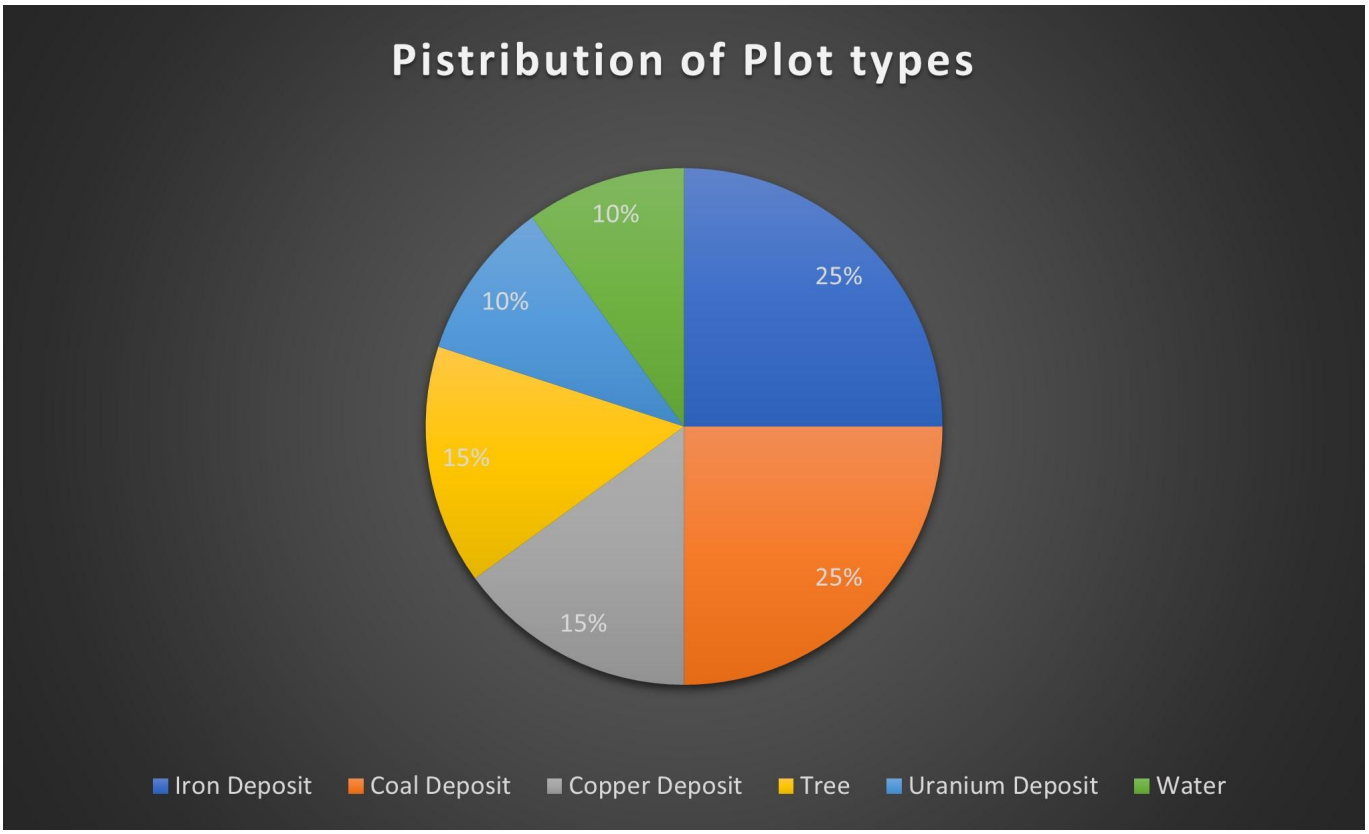
Pour rendre notre jeu le plus interactif et intéressant pour l'utilisateur nous avons dû imaginer un gameplay. La consigne principale est que ce gameplay soit axé autour de la gestion de l'électricité tout en maximisant la satisfaction des habitants pour être le plus efficace possible.

i. Les Ressources

Le système de ressources que nous avons mis en place est assez simple. Chaque ressource est composée d'une qualité et de son type. Il existe les types de ressources suivants : WOOD, COAL, IRON, WATER, FOOD, COPPER, URANIUM, SATISFACTION. La satisfaction est la ressource que l'on pourrait assimiler à un score, l'objectif final du jeu pour l'utilisateur est de produire un maximum de satisfaction et pour se faire il doit utiliser les bâtiments. Toutes les ressources sont

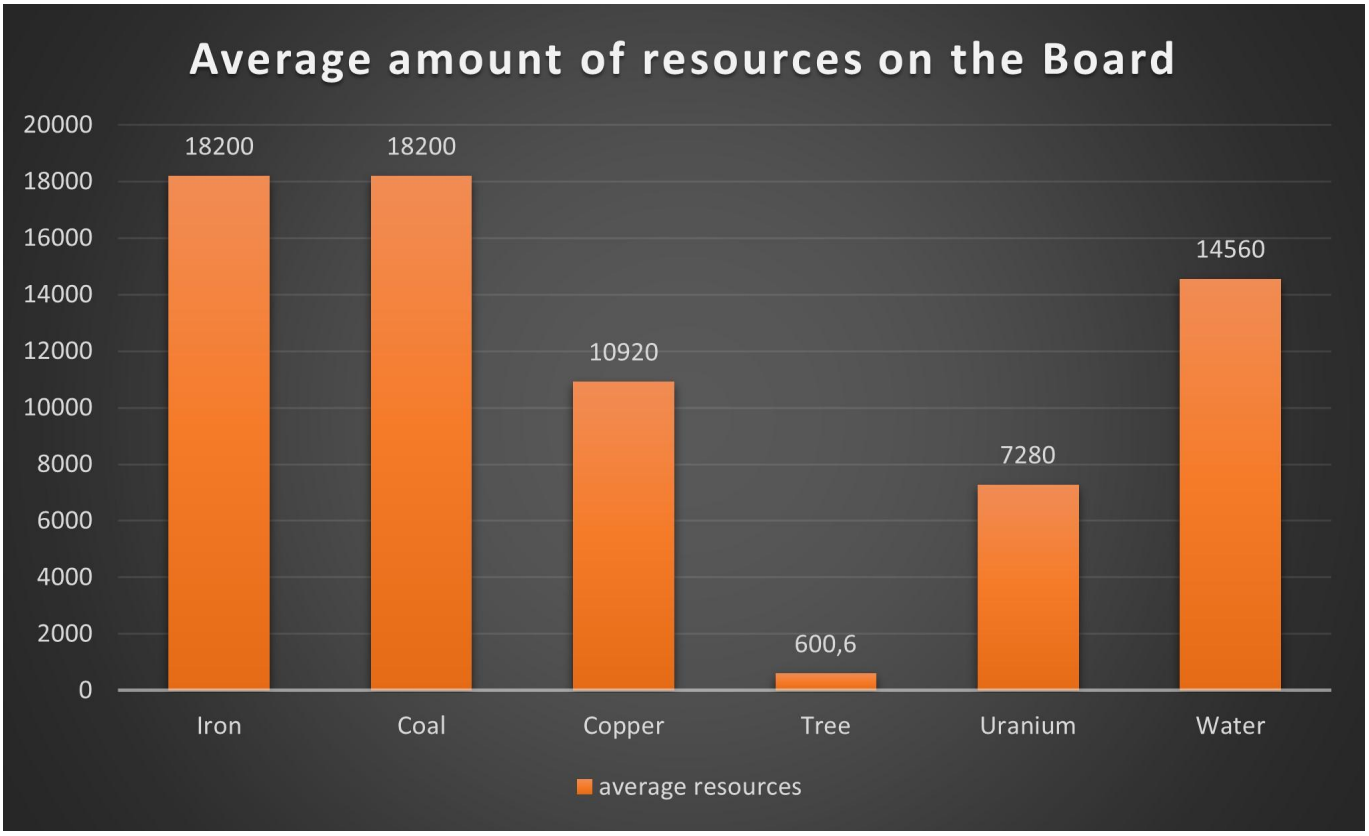
ii. Génération de la Carte

Sur la carte de jeu ou Board, chaque case peut contenir un bâtiment et/ou des ressources souterraines. Lors de la génération de la carte en début de partie, il existe six types de casse possible : une case d'eau, un gisement de fer, un gisement de cuivre, un gisement de charbon, un gisement d'uranium ou une case d'herbe avec un arbre. Pour chaque case son type est choisi aléatoirement lors de la génération en suivant la répartition suivante :



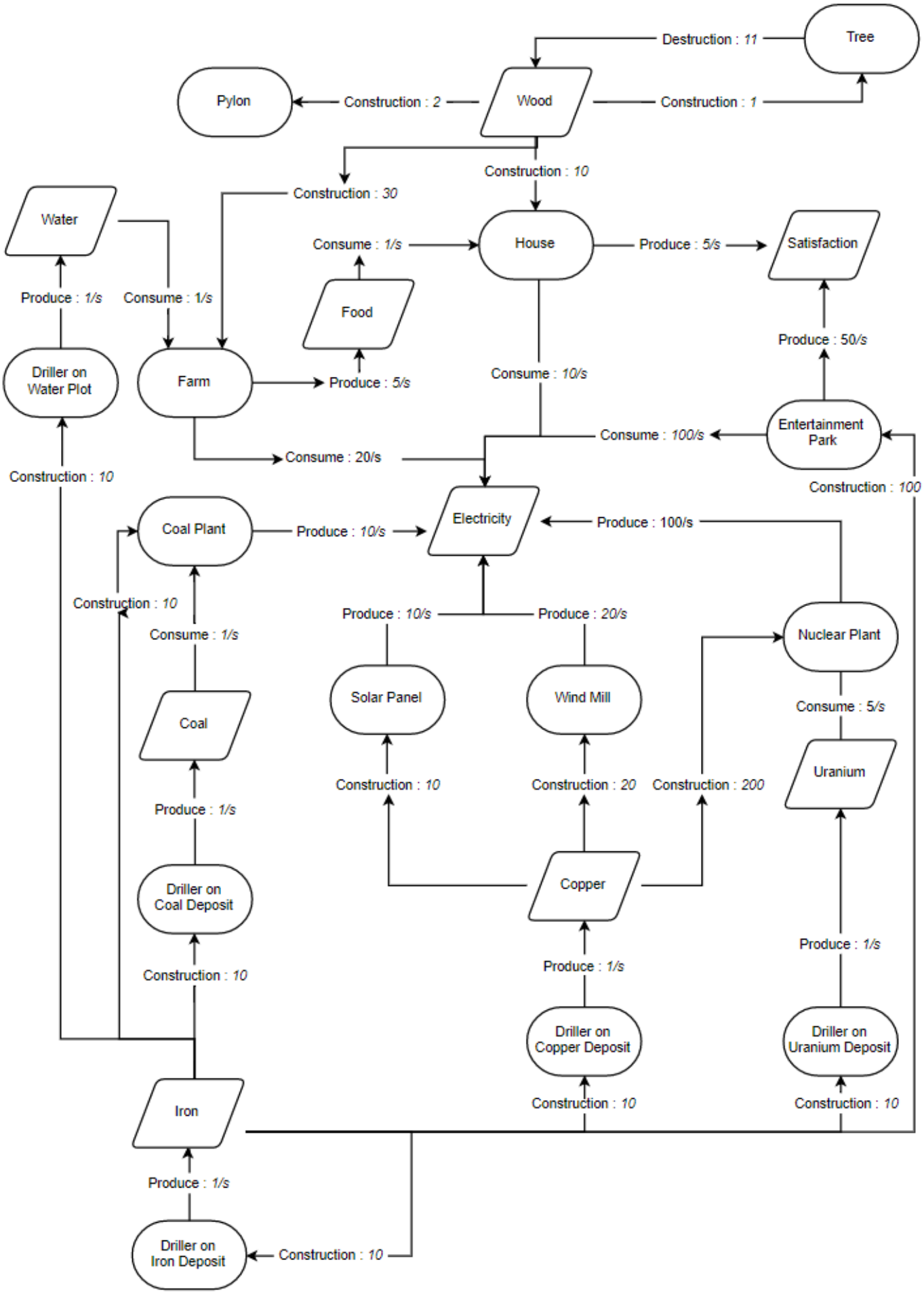
Nous avons notamment fait ces choix d'équilibrage en fonction de l'utilité en jeu de ces ressources, en effet le fer et le charbon étant les ressources les plus importantes du jeu, c'est pour ça que l'on en retrouve beaucoup. Chaque gisement et case contenant de l'eau contiennent une quantité fixée et finie de ressources disponibles. De plus, la carte de jeu à une taille finie, elle aussi, c'est pourquoi on peut calculer la quantité moyenne de ressources que l'on peut espérer trouver dans une partie. Sachant qu'une carte est composée de 26 case par

14 on a alors 364 case sur toute la carte et les gisements miniers contiennent 200 ressources et l'eau 400, les arbres quant à eux contiennent 11 de bois, voila les ressources attendues :



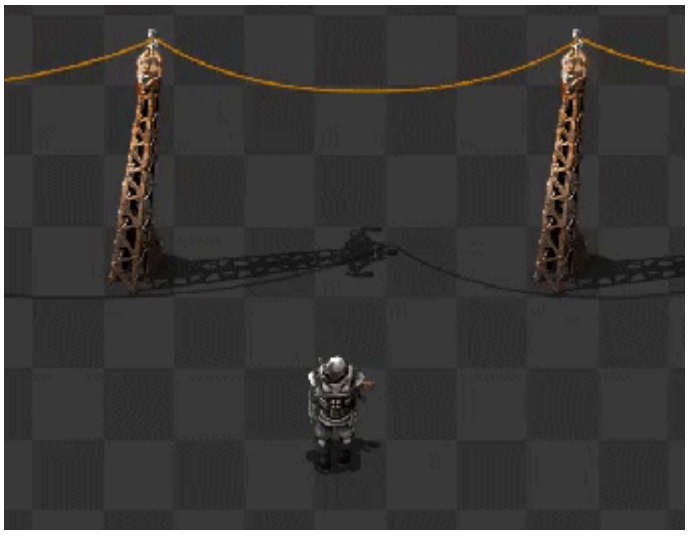
iii. Les Bâtiments

Les bâtiments sont le cœur du gameplay de notre jeu, ils peuvent être construits sur une case de la carte. Il existe plusieurs types de bâtiment différents que voici : TREE, PYLON, ROAD, HOUSE, FARM, ENTERTAINMENT_PARK, NUCLEAR_PLANT, COAL_PLANT, WINDMILL, SOLAR_PANEL, DRILLER. Chaque bâtiment est notamment défini par son coût de construction, ce qu'il consomme (électricité et ressources), ce qu'il produit et la qualité de pollution qu'il rejette. Lors de leur destruction, le joueur récupère dans son inventaire la totalité des ressources qui ont été utilisées pour le construire. Néanmoins, ceci n'est pas le cas pour les arbres qui coûtent 1 de bois à construire et donnent 11 bois lors de leur destruction. Les interactions entre les bâtiments et les ressources sont assez complexes, voici donc un diagramme qui détail cela :


Tree
Destruction : 11
Pylon
Construction : 2
Wood
Construction : 1
Construction : 10
Construction : 30
Water
Consume : 1/s
House
Produce : 5/s
Satisfaction
Produce : 1/s
Consume : 1/s
Food
Produce : 50/s
Driller on Water Plot
Farm
Produce : 5/s
Consume : 10/s
Entertainment Park
Consume : 100/s
Consume : 20/s
Construction : 10
Construction : 100
Coal Plant
Produce : 10/s
Electricity
Produce : 100/s
Construction : 10
Produce : 10/s
Produce : 20/s
Nuclear Plant
Consume : 1/s
Solar Panel
Wind Mill
Consume : 5/s
Coal
Uranium
Construction : 10
Construction : 20
Construction : 200
Produce : 1/s
Copper
Driller on Coal Deposit
Produce : 1/s
Construction : 10
Produce : 1/s
Driller on Copper Deposit
Driller on Uranium Deposit
Iron
Construction : 10
Construction : 10
Produce : 1/s
Driller on Iron Deposit
Construction : 10

iv. Les Réseaux électriques

Pour mettre en place les réseaux électriques dans ce jeu, nous avons décidé de nous inspirer de ce qui est fait dans le jeu de gestion et d'automatisation Factorio. Les pylônes sont les constructions qui transportent l'électricité. Un réseau électrique est constitué de pylônes et de bâtiments. Deux pylônes font partie du même réseau s'ils sont à une distance inférieure ou égale à quatre cases l'un de l'autre. Un bâtiment fait partie d'un réseau s'il est directement autour d'un pylône. Chaque réseau possède une quantité d'électricité disponible qui varie en fonction de ce qui est produit et consommé sur le réseau. S'il n'y a pas suffisamment d'électricité disponible sur le réseau, les bâtiments qui en ont besoin ne sont alors pas en capacité de produire leur ressources.



Les pylons sur Factorio

v. Les Réseaux routiers

En ce qui concerne les réseaux routiers, nous avons choisi de rajouter un certain nombre de contraintes pour le joueur. Tout d'abord, la carte générée au départ possède un certain nombre de routes qui sont toutes liées. En effet, une règle importante concernant les routes est qu'une route doit être adjacente à une autre. On se retrouve donc avec un seul grand réseau plutôt qu'une collection de petits réseaux, ce qui est un choix arbitraire. L'utilité du réseau routier est de pouvoir construire des bâtiments autour des routes, comme il s'agit de la seule manière d'agrandir son territoire.

Sprite de route

vi. Les Excavateurs

Afin de récupérer les ressources minières ainsi que l'eau, on place sur les cases de la carte correspondante le bâtiment nommé Driller. Ce bâtiment à pour fonction de prélever les ressources contenues dans le sol et de les ajouter à l'inventaire du joueur. Les ressources contenues dans le gisement étant finies, une fois qu'il n'y en a plus, la case change de type et devient une case d'herbe. Ce qui ajoute une dimension supplémentaire dans le choix de stratégie du joueur.

f. L'interface graphique

Lors de la phase de conception le fonctionnement exact de l'interface graphique était un peu flou, car nous n'avions jamais concrètement travaillé avec, nous savions cependant comment les différentes données à afficher aller être récupérées. Nous avons fait le choix d'utiliser JavaFX et sceneBuilder pour réaliser notre interface graphique afin d'avoir un visuel plus agréable, et nous avons déterminé durant la phase de conception que seul la class "Map" communiquerais les informations à afficher avec les controller des différentes scènes et qu'il y aurait deux scènes: le plateau de jeu et le menu principal.

3. Phase de développement

a. Outil de développement

L'entièreté du projet a été réalisée en Java à l'aide de l'IDE IntelliJ, étant 4 pour réaliser ce projet, nous avons utilisé un repository github pour partager le code et développer le jeu en collaboration. Pour ce qui est de l'interface graphique, nous avons décidé d'utiliser Javafx et SceneBuilder. Nous avons fait le choix d'utiliser Maven, ce qui nous a permis de pouvoir facilement importer différentes extensions.

Logo de IntelliJ IDEA

b. Méthode de travail

Après la création des différents diagrammes et la rédaction du rapport de conception, nous avons commencé le développement en se reposant sur le travail effectué précédemment

c. Répartition des tâches

Nous avons divisé notre groupe en deux équipes, une pour première pour créer les fonctionnalités du jeu en lui-même (génération de la carte, créations des bâtiments, production de ressources…) et une seconde pour toute la partie graphique (affichage de la carte, affichage des bâtiments, création des menus…)

d. Fonctionnalités implémentées

- Menu principal
- Interface graphique
- Fonctionnalité de sauvegarde
- Gestion d'électricité
- Différents types de ressources
- Différents types de bâtiments
- Différents types de terrain

e. Fonctionnalité de sauvegarde

L'utilisateur à la possibilité de sauvegarder sa partie, cette fonctionnalité ajoute une meilleure qualité de vie au programme. La partie peut être chargée à partir du menu principal. On appel alors dans ce cas le constructeur de la classe Map en passant comme paramètre le fichier de sauvegarde.Un fichier de sauvegarde se présente de la manière suivante:

```
5,5
Coal:94.0,Iron:0.0,Copper:100.0,Uranium:0.0,Oil:0.0,Water:0.0,Gas:0.0,Wood:100.0,Euros:0.0,Satisfaction:0.0
CLAY,false,Ro,Coal:28.0
GRASS,false,Tr,null:0
CLAY,false,Ro,Iron:65.0
SAND,false,Ro,null:0
WATER,false,null,Water:11.0
WATER,false,null,Water:51.0
GRASS,false,Tr,null:0
SAND,true,null,null:0
STONE,true,null,null:0
CLAY,true,null,null:0
STONE,true,null,Coal:85.0
SAND,true,null,null:0
DIRT,true,null,null:0
GRASS,false,Tr,null:0
SAND,true,null,Gas:65.0
GRASS,false,Tr,null:0
DIRT,true,null,null:0
STONE,true,null,Copper:27.0
WATER,false,null,Water:17.0
CLAY,true,null,null:0
DIRT,true,null,null:0
GRASS,false,Tr,Iron:76.0
STONE,true,null,Coal:73.0
DIRT,true,null,Oil:12.0
GRASS,false,Tr,null:0
```

Fichier de sauvegarde

f. Interface Graphique

Nous avons choisis de réaliser notre interface graphique à l'aide de javaFX, le résultat est assez proche de nos attente lors de la phase de conception, nous avons 2 fichier .fxml MainMenu et GameBoard, générer via scenebuilder, le premier sert à afficher le menu principale et à proposer au joueur s'il veut lancer une nouvelle partie, charger une partie existante ou quitter le jeu via des boutons, le second est l'affichage du jeu lui même permettant au joueur de voir la carte et les interfaces lui permettant de construire et détruire des bâtiments. Nous avons également un troisième fichier .fxml InGameMenu qui n'était pas prévu dans la phase de conception permettant au joueur de sauvegarder sa partie et de quitter le jeu.

4. Conclusion

Pour conclure, durant ce projet, nous avons développé un jeu fonctionnel en Java, répondant au cahier des charges, tout cela en travaillant en équipe. Durant ce projet, nous nous sommes rendu compte de l'importance de la réalisation de diagramme et d'étude préventive avant d'entamer la phase de développement, en effet, même si les diagrammes réaliser avant le développement ne correspondent pas exactement à la réalité, ils ont permis d'envisager la façon de concevoir le projet de manière plus claire et précise et donc de faciliter le développement. Enfin nous avons ajouté plusieurs fonctionnalités intéressantes, tel que la possibilité de sauvegarder une partie en cours ou le transfert de l'électricité en réseau, nous avons également conçu une interface graphique décente à l'aide de javaFX, et cela, dans un mode de fonctionnement projet semblable à ce que l'on pourrait trouver en entreprise. Nous avons pu également mettre en pratique les différentes notions vues à travers l'UV.