



Projet Approche Objet

Abdoulaye Diallo, Théophyl Hoguet, Kea Horvath

Décembre 2023

Table des matières

1	Analyse	3
1.1	Présentation	3
1.2	Bâtiments	3
1.3	Comment jouer	4
1.3.1	But du jeu :	4
1.3.2	Début du jeu :	4
1.3.3	Condition de perte :	4
1.3.4	Obtention/ Perte de ressources :	5
1.3.5	Construction/Destruction de bâtiments :	5
1.3.6	Gestion du temps :	5
1.3.7	Score :	5
1.3.8	Interface Utilisateur :	5
2	Conception	6
2.1	DDD + Architecture de nos classes	6
2.2	Début d'une partie	7
2.3	Options de notre jeu	8
2.4	Interface graphique	8
2.4.1	Voici les fonctionnalités de notre interface graphique :	8
3	Conclusion	10
4	Annexe-	10

1 Analyse

1.1 Présentation

Pour ce projet de jeu de stratégie nous nous sommes mis à trois : Abdoulaye Diallo, Théophyl Hoguet et Kea Horvath. Le but du jeu est de permettre au joueur de gérer de manière stratégique une ville virtuelle, en prenant en charge la gestion des ressources, des habitants, des travailleurs et des bâtiments. L'objectif ultime est de faire prospérer l'économie de la ville aussi longtemps que possible. Le joueur a la possibilité d'utiliser des ressources pour construire de nouveaux bâtiments. Chaque bâtiment créé consomme une certaine quantité de ressource mais peut également en fabriquer. De plus, les habitants et les travailleurs consomment de la nourriture (une unité de FOOD par jour et par habitant). Le joueur doit prendre des décisions réfléchies pour maximiser la production, minimiser les pertes, et maintenir l'équilibre financier de la ville. Le joueur perd quand il n'y a plus nourriture (FOOD) pour nourrir sa population.

1.2 Bâtiments

Les bâtiments que nous avons inclus dans notre implémentation sont :

- **Wooden Cabin :**
 - Abrite : 4 habitants + 4 travailleurs minimum
 - Coûte : 1 GOLD + 1 WOOD
 - Prend 2 jours à construire
 - Produit : 6 WOOD + 8 FOOD
- **House :**
 - Abrite : 8 habitants minimum
 - Coûte : 1 GOLD + 2 WOOD + 2 STONE
 - Prend 4 jours à construire
 - Produit : rien
- **Apartment Building :**
 - Abrite : 32 habitants minimum
 - Coûte : 4 GOLD + 50 WOOD + 50 STONE
 - Prend 6 jours à construire
 - Produit : rien
- **Farm :**
 - Abrite : 10 habitants + 5 travailleurs minimum
 - Coûte : 4 GOLD + 5 WOOD + 5 STONE
 - Prend 2 jours à construire
 - Produit : 10 FOOD
- **Quarry :**
 - Abrite : 5 habitants + 15 travailleurs minimum
 - Coûte : 4 GOLD + 50 WOOD
 - Prend 2 jours à construire

- Produit : 4 STONE + 4 IRON + 4 COAL + 2 GOLD
- **Lumber Mill :**
 - Abrite : 10 travailleurs minimum
 - Coûte : 6 GOLD + 50 Wood + 50 Stone
 - Prend 4 jours à construire
 - Consomme : 4 WOOD
 - Produit : 4 LUMBER
- **Cement Plant :**
 - Abrite : 10 travailleurs minimum
 - Coûte : 6 GOLD + 50 Wood + 50 Stone
 - Prend 4 jours à construire
 - Consomme : 4 COAL + 4 CEMENT
 - Produit : 4 CEMENT
- **Steel Mill :**
 - Abrite : 20 travailleurs minimum
 - Coûte : 6 GOLD + 100 Wood + 50 Stone
 - Prend 6 jours à construire
 - Consomme : 2 COAL + 4 STEEL
 - Produit : 4 STEEL
- **Tool Factory :**
 - Abrite : 12 travailleurs minimum
 - Coûte : 8 GOLD + 50 Wood + 50 Stone
 - Prend 8 jours à construire
 - Consomme : 4 COAL + 4 STEEL
 - Produit : 4 TOOL

1.3 Comment jouer

1.3.1 But du jeu :

Le but du jeu est de maintenir une économie stable le plus longtemps possible sans que la nourriture ne vienne à manquer. Le joueur doit prendre garde à toujours avoir assez de ressources pour nourrir tout sa population.

1.3.2 Début du jeu :

Au début du jeu le joueur possède des ressources initiales et un ou deux bâtiments. Le joueur commence sans habitants ou travailleurs, il pourra ajouter autant d'habitants et/ou travailleurs que la capacité du bâtiment le permettra.

1.3.3 Condition de perte :

La partie se termine si la quantité de nourriture restante n'est pas suffisante pour tous les habitants et travailleurs de la ville. Donc la quantité de

nourriture disponible représente la satisfaction de la population de la ville.

1.3.4 Obtention/ Perte de ressources :

Le joueur obtient des ressources chaque nouveau jour en fonction des bâtiments qui en produisent. La plupart des bâtiments produisent des ressources collectées en début de journée. Le joueur perd une quantité de nourriture proportionnelle au nombre d'habitants et de travailleurs dans la ville. Il a également la possibilité d'utiliser ces ressources pour construire des bâtiments.

Dans notre jeu, il est possible d'améliorer les bâtiments pour un certain coût, en fonction du bâtiment et de son niveau. A chaque niveau le coût d'amélioration du bâtiment augmente mais ses productions aussi.

1.3.5 Construction/Destruction de bâtiments :

Pour construire un bâtiment, le joueur doit utiliser ses ressources. Chaque bâtiment a un coût en or et en matériaux qui lui est propre. De plus, chaque bâtiment a une durée de construction avant d'être disponible et de produire ses ressources. A l'opposé, la destruction d'un bâtiment se fait immédiatement. Le joueur récupère alors la moitié des ressources qu'il a utilisé pour construire le bâtiment, on ignore les améliorations faites au bâtiment.

1.3.6 Gestion du temps :

Dans notre jeu une journée dure une seconde et demi. Ceci pourrait sembler rapide mais le joueur a l'option de mettre le jeu en pause pour prendre ses décisions.

1.3.7 Score :

Nous avons aussi décidé de mettre en place un compteur de score qui évolue en fonction du nombre de ressource et d'un multiplicateur de ressource.

1.3.8 Interface Utilisateur :

Nous avons choisi d'implémenter une interface graphique pour notre jeu. La ville est représentée par un grid de plusieurs cellules, qui seront les emplacements des bâtiments construits. Pour construire un bâtiment, le joueur clique gauche sur un bâtiment et sur une cellule libre.

2 Conception

2.1 DDD + Architecture de nos classes

Pour la partie conception de notre projet nous avons décidé d'utiliser le DDD(Domain Driven Design). Le DDD est le design pattern le plus utile pour un jeu avec une division de l'application en plusieurs couches. Ce qui simplifie grandement la conception et la maintenance du projet.

Dans le cadre du DDD, nous identifions plusieurs **Value Objects** importants pour la représentation des concepts clés de notre jeu. Parmi eux, nous avons les objets tels que **BuildingType** pour décrire les différents types de bâtiments, ainsi que des objets relatifs aux ressources tels que **Resource**, **ResourceType**, **ResourceList**, et **ResourceAmount**. Nous incorporons également des objets tels que **Grid** pour la gestion de l'espace dans le jeu, ainsi que des objets liés aux besoins tels que **Needs**, et **Production**, **Consumption**, et **ConstructionNeeds** qui étendent **Needs**.

Dans notre jeu, nous avons **Building** comme une **Entity** principale, représentant les bâtiments dans notre ville. Enfin, l'**Aggregate Manager** est présente pour englober et coordonner les différents éléments du jeu, assurant ainsi une gestion efficace et une cohérence globale. Notre **Manager** nous permet de manipuler tout les aspects de notre jeu grâce aux **Value Objects** de nos différents bâtiments.

Les classes qui manipulent le plus notre jeu sont **Manager** et **Building**, elles permettent de :

- Construire/ Détruire des bâtiments : `buildBuilding()` / `destroyBuilding()` dans **Manager**
- Ajouter / Supprimer des habitants/travailleurs : `addInhabitantsToBuilding()` / `removeInhabitantsFromBuilding()` dans **Manager**
- Mettre le jeu en pause ou le terminer si le joueur n'a plus de nourriture :
- Mettre à jour toutes ressources du jeu en fonction des consommations et des productions : `update()` dans **Manager**
- Améliorer des bâtiments : `upgradeBuilding()` dans **Manager**

2.2 Début d'une partie

Au début du jeu, les joueurs peuvent choisir l'un des trois niveaux de difficulté proposés : **Easy**, **Normal**, ou **Hard**. Ce choix est représenté par la classe enum **GameStarter**. Chaque niveau de difficulté est associé à un ensemble de ressources et de bâtiments initiaux. Par exemple, le niveau **Easy** offre plus de ressources que les niveaux **Normal** et **Hard**. La classe **GameStarter** stocke ces informations. Si le joueur choisit le niveau de difficulté **Easy**, la classe **Main** utilise les informations définies dans l'énumération **GameStarter.EASY** pour initialiser la partie. Ces informations déterminent le nombre d'habitants, de travailleurs, les ressources de départ et les bâtiments avec lesquelles le joueur commence. L'utilisation d'une énumération permet de gérer facilement les différentes configurations de jeu.

Une fois le niveau choisi le joueur peut créer des bâtiments pour gagner plus de ressources, ou rajouter des habitants ou travailleurs dans les bâtiments. Par exemple, si le joueur choisit de construire un bâtiment **HOUSE**, **App** va appeler sa méthode **buildBuilding** qui va appeler **buildBuilding** de **Manager**. **Manager** va vérifier si l'emplacement choisi pour la construction du bâtiment est libre. Si elle l'est le bâtiment sera construit, sinon une **exception** sera lancé. Le joueur peut construire autant de bâtiments qu'il le souhaite tant qu'il a les ressources pour ces constructions. Une fois les constructions faites le joueur peut ajouter des habitants ou travailleurs qui consommeront des ressources. Si le joueur a assez de ressources il peut améliorer ses bâtiments déjà construits. Ceci appelle **isBuildingUpgradeable** de **Manager** qui appelle **canUpgrade** de **Building** pour vérifier les ressources du joueur. Si **canUpgrade** retourne **true** alors **upgrade** de **Building** est lancé pour améliorer le bâtiment. Une fois amélioré le bâtiment produit et consomme plus de ressources. Le joueur pourra continuer à construire et/ou modifier ses bâtiments ou encore augmenter sa population tant qu'il possède des ressources.

2.3 Options de notre jeu

- **Play / Pause** : Notre jeu offre au joueur la possibilité de mettre le jeu en pause et de le reprendre ensuite à l'aide d'un bouton. Cela donne un temps de réflexion au joueur.
- **Construction d'un bâtiment** : Le joueur choisit le bâtiment qu'il souhaite construire, et dont il a les ressources. Il choisit ensuite une cellule libre dans laquelle cet bâtiment sera placé.
- **Amélioration des bâtiments** : Lorsque le joueur souhaite améliorer un bâtiment
 - Chaque amélioration augmente les coûts de construction de 25% pour la prochaine amélioration.
 - Chaque amélioration double la consommation et la production du bâtiment.
 - Chaque amélioration multiplie le nombre d'habitants et/ou de travailleurs par 1.5.
- **Destruction d'un bâtiment** : Lorsque le joueur souhaite détruire un bâtiment, il effectue un clic droit sur la cellule du bâtiment, un menu se déroule, et il peut ensuite cliquer sur "Destroy."
- **Ajout / Suppression d'habitants / de travailleurs** : Un clic-droit sur un bâtiment déroule un menu avec les options relatives aux habitants/travailleurs du bâtiment.
- **Sauvegarde d'une partie dans un fichier** : À tout moment pendant sa partie, le joueur peut mettre le jeu en pause et sauvegarder dans un fichier.
- **Chargement d'une partie à partir d'un fichier** : Le joueur a le choix de créer une nouvelle partie ou de jouer à partir d'une partie précédemment sauvegardée.

2.4 Interface graphique

Pour l'interface utilisateur nous avons décidé de créer une interface graphique avec javafx.

2.4.1 Voici les fonctionnalités de notre interface graphique :

- Bouton Play / Pause
- Construction de bâtiment : faire un clic-gauche sur un bâtiment que l'on veut créer puis faire un clic-gauche sur une cellule libre du grid.
- Modifications de bâtiments : faire un clic-droit sur un bâtiment déroule un menu de choix possibles.
 - Destruction de bâtiment : choisir "Destroy"
 - Amélioration de bâtiment : choisir "Upgrade"
 - Ajouter des habitants : choisir "Add Inhabitants"
 - Supprimer des habitants : choisir "Remove Inhabitants"

- Ajouter des travailleurs : choisir "Add Workers"
- Supprimer des travailleurs : choisir "Remove Workers"
- Affichage des informations d'un bâtiment : faire un clic-gauche sur un bâtiment déjà construit

3 Conclusion

En conclusion, durant ce projet nous avons pu voir les points délicats de la création de jeu de stratégie et les décisions qui poussent à incorporer un aspect plutôt qu'un autre.

L'utilisation du DDD nous a permis de travailler de manière efficace et de diviser le projet en plusieurs sous-domaines qui ont rendu le tout très cohérent. Les Value Objects, Entities, et Aggregates définis par le DDD ont été essentiels pour représenter efficacement les concepts clés du jeu, tels que les types de bâtiments et les ressources.

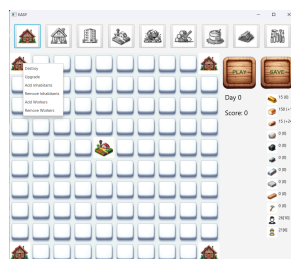
En plus de la conception, notre ajout d'une interface graphique rend notre jeu plus agréable à jouer. Nous pensons que les ajouts et modifications de bâtiments se font facilement.

En fin de compte, ce projet nous a permis de consolider nos compétences en programmation orientée objet, la réflexion derrière la conception de jeu et la collaboration en équipe. Bien que le jeu soit assez simple en termes d'options, nous considérons que nous avons accompli les grandes attentes de notre jeu.

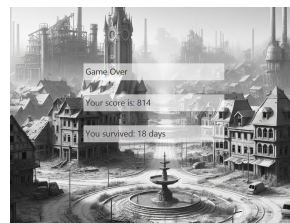
4 Annexe-



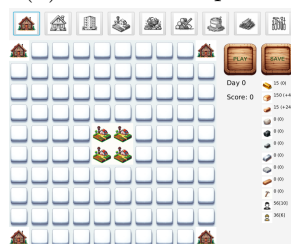
(a) Lancement d'une partie



(b) En cours de partie



(c) Fin de partie



(d) Fin de partie

TABLE 1 – Images de notre jeu