Projet Approche Objet

CHELH MONIR

M1 Informatique

2025-2024

Ce projet, réalisé individuellement, représente pour moi un défi personnel que j'ai souhaité relever. J'ai volontairement choisi de mener ce travail seul, non par nécessité, mais par envie d'explorer pleinement mes capacités et de m'investir totalement dans chaque étape de sa conception et de sa réalisation. Cette démarche m'a permis d'acquérir une maîtrise approfondie des différents aspects du projet et de développer une plus grande autonomie. Conscient de l'ampleur de la tâche entreprise en solitaire, je vous prie de bien vouloir tenir compte de ce choix et de m'excuser par avance pour toute éventuelle imperfection ou lacune qui pourrait en résulter.

Table des matières

1. Analyse :

1.1. Présentation :

L'objectif principal du jeu est de permettre au joueur de gérer une ville virtuelle de manière stratégique. Cela inclut la gestion des ressources, des habitants, des travailleurs et des bâtiments. Le but ultime est d'assurer la prospérité économique de la ville aussi longtemps que possible.

Le joueur peut utiliser les ressources disponibles pour construire de nouveaux bâtiments. Chaque bâtiment consomme une certaine quantité de ressources lors de sa construction, mais peut également en produire par la suite. En parallèle, les habitants et les travailleurs nécessitent de la nourriture (une unité de FOOD par jour et par habitant).

Le joueur doit prendre des décisions stratégiques pour optimiser la production, limiter les pertes et maintenir un équilibre financier stable. La partie se termine si le joueur n'a plus suffisamment de nourriture (FOOD) pour nourrir la population.

1.2. Bâtiments :

Les bâtiments intégrés dans notre implémentation sont les suivants :

	Espace sur le plan (x × y)	Nombre d'habitants	Nombre de travailleurs	Matériaux de construction	Consommation (/jour)	Production (/jour)	Temps de construction (/jour)
`WoodenCabin`	1×1	2	2	1 Wood	∅	Wood 2, Food 2	2
`House`	2×2	4	0	2 Wood, 2 Stone	∅	∅	4
`ApartmentBuilding`	3×2	60	0	50 Wood, 50 Stone	∅	∅	6
`Farm`	3×3	5	3	5 Wood, 5 Stone	∅	Food 10	2
`Quarry`	2×2	2	30	50 Wood	∅	Stone 4, Iron 4, Coal 4	2
`LumberMill`	3×3	0	10	50 Wood, 50 Stone	Wood 4	Lumber 4	4
`CementPlant`	4×3	0	10	50 Wood, 50 Stone	Stone 4, Coat 4	Cement 4	4
`SteelMill`	4×3	0	40	100 Wood, 50 Stone	Iron 4, Coal 2	Steel 4	6
`ToolFactory`	4×3	0	12	50 Wood, 50 Stone	Steel 4, Coal 4	Tool 4	8

1.3. Comment Jouer :

1.3.1. But et Début du jeu :

Le but du jeu est de maintenir une économie stable aussi longtemps que possible tout en évitant une pénurie de nourriture, essentielle pour nourrir toute la population. Le joueur doit gérer efficacement les ressources afin d'assurer la prospérité de la ville.

Au début de la partie, le joueur dispose de ressources initiales ainsi que d'un ou deux bâtiments. Il commence sans habitants ni travailleurs, mais pourra en ajouter en fonction de la capacité des bâtiments disponibles. Une gestion stratégique sera nécessaire pour développer la ville tout en veillant à ne pas manquer de ressources alimentaires.

1.3.2. Condition de perte :

La partie se termine lorsque la quantité de nourriture disponible devient insuffisante pour nourrir l'ensemble des habitants et des travailleurs de la ville. Ainsi, la disponibilité de nourriture reflète directement le niveau de satisfaction de la population.

1.3.3. Obtention et Pertes des ressources :

Chaque nouveau jour, le joueur obtient des ressources en fonction des bâtiments de production présents dans la ville. La plupart des bâtiments génèrent des ressources collectées au début de la journée. Cependant, le joueur perd également une quantité de nourriture proportionnelle au nombre d'habitants et de travailleurs.

Les ressources accumulées peuvent être utilisées pour construire de nouveaux bâtiments, contribuant ainsi au développement de la ville.

Dans notre jeu, il est également possible d'améliorer les bâtiments moyennant un certain coût, qui varie en fonction du type de bâtiment et de son niveau actuel. À chaque amélioration, le coût augmente, mais la production du bâtiment s'accroît proportionnellement, offrant un meilleur rendement au fil des niveaux.

1.3.4. Construction/Destruction de bâtiments :

Pour construire un bâtiment, le joueur doit dépenser ses ressources. Chaque bâtiment possède un coût spécifique en or et en matériaux. De plus, chaque construction nécessite un temps défini avant que le bâtiment ne soit opérationnel et commence à produire des ressources.

En revanche, la destruction d'un bâtiment est immédiate. Lors de cette opération, le joueur récupère la moitié des ressources initialement utilisées pour la construction, sans tenir compte des améliorations effectuées sur le bâtiment.

1.3.5. Gestion du temps :

Dans notre jeu, une journée équivaut à une seconde et demie. Bien que ce rythme puisse sembler rapide, le joueur dispose de l'option de mettre le jeu en pause pour prendre le temps de réfléchir et prendre ses décisions stratégiques.

1.3.6. Score et Interface Utilisateur :

Nous avons décidé d'intégrer un système de score qui évolue en fonction de la quantité de ressources accumulées et d'un multiplicateur de ressource. Ce mécanisme permet de mesurer la performance du joueur tout au long de la partie.

Concernant l'interface utilisateur, nous avons opté pour une représentation graphique intuitive. La ville est organisée sous forme d'une grille composée de plusieurs cellules, correspondant aux emplacements disponibles pour les bâtiments. Pour construire un bâtiment, le joueur peut effectuer un clic gauche sur le bâtiment souhaité, puis sélectionner une cellule libre sur la grille.

2. Conception :

2.1. Choix D'Architecture de notre Projet :

Pour la conception de notre projet, nous avons choisi d'adopter le **Domain-Driven Design (DDD)**. Ce design pattern est particulièrement adapté à un jeu nécessitant une division en plusieurs couches, car il simplifie à la fois la conception et la maintenance du projet.

Dans le cadre du DDD, nous avons identifié plusieurs *Value Objects* essentiels pour représenter les concepts clés de notre jeu. Parmi eux, nous avons :

- **BuildingType** : décrit les différents types de bâtiments.
- Des objets relatifs aux ressources, tels que **Resource**, **ResourceType**, **ResourceList**, et **ResourceAmount**.
- **Grid** : pour gérer l'espace de la ville.
- Des objets liés aux besoins et à la production, comme **Needs**, et ses extensions : **Production**, **Consumption**, et **ConstructionNeeds**.

Dans notre jeu, l'entité principale est **Building**, qui représente les bâtiments de la ville. Par ailleurs, nous utilisons un **Aggregate Manager**, qui regroupe et coordonne les différents éléments du jeu. Ce Manager assure une gestion efficace et garantit la cohérence globale. Il permet également de manipuler tous les aspects du jeu grâce aux *Value Objects* associés aux 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 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 ont le choix entre trois niveaux de difficulté : *Easy*, *Normal*, ou *Hard*. Ce choix est représenté par la classe énumérée **GameStarter**. Chaque niveau de difficulté est associé à un ensemble spécifique de ressources et de bâtiments de départ. Par exemple, le niveau **Easy** offre davantage de ressources que les niveaux **Normal** et **Hard**. Ces informations sont stockées dans la classe **GameStarter**.

Si le joueur choisit le niveau **Easy**, la classe **Main** utilise les données définies dans **GameStarter.EASY** pour initialiser la partie, déterminant ainsi le nombre d'habitants, de travailleurs, les ressources initiales et les bâtiments avec lesquels il commence. L'utilisation d'une énumération facilite la gestion des différentes configurations de jeu.

Une fois le niveau choisi, le joueur peut commencer à construire des bâtiments pour générer plus de ressources ou ajouter des habitants et travailleurs. Par exemple, si le joueur choisit de construire un bâtiment **HOUSE**, l'application appelle sa méthode **buildBuilding**, qui à son tour appelle **buildBuilding** dans le *Manager*. Ce dernier vérifie si l'emplacement sélectionné est libre. Si l'emplacement est disponible, le bâtiment est construit, sinon une exception est lancée. Le joueur peut construire autant de bâtiments qu'il souhaite, tant qu'il dispose des ressources nécessaires.

Après la construction des bâtiments, le joueur peut y ajouter des habitants ou des travailleurs, qui consommeront des ressources. S'il a suffisamment de ressources, il peut améliorer les bâtiments existants. Cela appelle la méthode **isBuildingUpgradeable** du **Manager**, qui invoque ensuite **canUpgrade** dans **Building** pour vérifier les ressources disponibles. Si **canUpgrade** retourne *TRUE*, l'amélioration du bâtiment est lancée via la méthode **upgrade** de **Building**. Une fois amélioré, le bâtiment produit et consomme davantage de ressources. Le joueur peut alors continuer à construire ou modifier ses bâtiments et augmenter sa population, tant qu'il dispose des ressources nécessaires.

2.3. Options de notre jeu :

- **Play / Pause** : Le jeu permet au joueur de mettre la partie en pause et de la reprendre via un bouton, offrant ainsi un temps de réflexion pour prendre des décisions stratégiques.
- **Construction d'un bâtiment** : Le joueur choisit le bâtiment qu'il souhaite construire, à condition d'avoir les ressources nécessaires. Ensuite, il sélectionne une cellule libre pour y placer le bâtiment.
- **Amélioration des bâtiments** : Lorsque le joueur désire améliorer un bâtiment :
  - ✓ Chaque amélioration augmente le coût de construction de 25% pour la prochaine amélioration.
  - ✓ Elle double la consommation et la production du bâtiment.
  - ✓ Elle multiplie le nombre d'habitants et/ou de travailleurs par 1,5.
- **Destruction d'un bâtiment** : Pour détruire un bâtiment, le joueur effectue un clic droit sur la cellule où se trouve le bâtiment, un menu s'affiche, et il peut cliquer sur "Destroy" pour le supprimer.
- **Ajout / Suppression d'habitants ou de travailleurs** : Un clic droit sur un bâtiment ouvre un menu permettant d'ajouter ou de retirer des habitants et travailleurs selon les besoins du joueur.
- **Sauvegarde d'une partie dans un fichier** : À tout moment pendant la partie, le joueur peut mettre le jeu en pause et sauvegarder sa progression dans un fichier.
- **Chargement d'une partie à partir d'un fichier** : Le joueur peut choisir de commencer une nouvelle partie ou de reprendre une partie précédemment sauvegardée.

2.4. Interface graphique

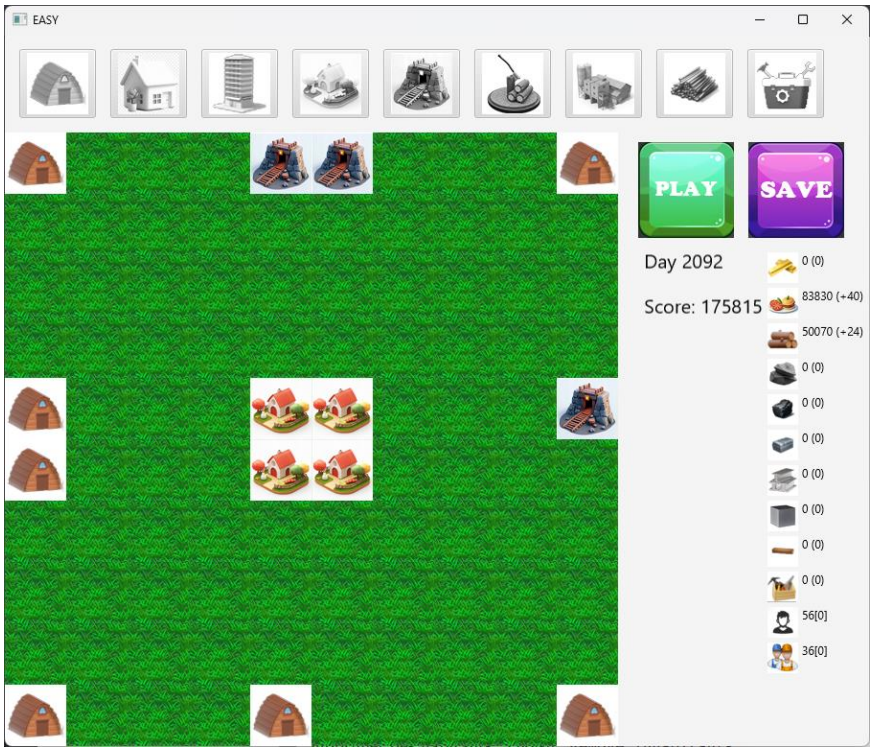
Pour l'interface graphique de notre jeu, nous avons opté pour ***JavaFX***.

Fonctionnalités de l'interface graphique :

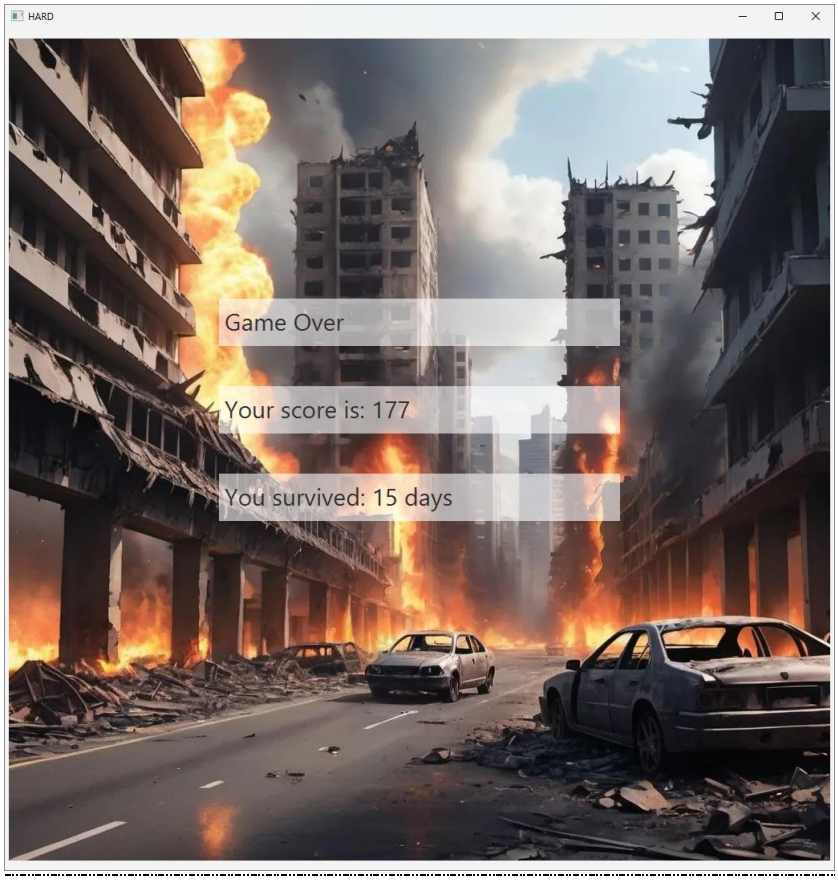
- **Bouton Play / Pause** : Permet de mettre le jeu en pause et de le reprendre.
- **Construction de bâtiment** : Un clic gauche sur un bâtiment à créer, suivi d'un clic gauche sur une cellule libre du grid pour le placer.
- **Modification de bâtiments** : Un clic droit sur un bâtiment affiche un menu avec les choix possibles.
- **Destruction de bâtiment** : Dans le menu déroulant, sélectionner l'option "Destroy" pour détruire un bâtiment.
- **Amélioration de bâtiment** : Sélectionner "Upgrade" dans le menu pour améliorer un bâtiment.
- **Ajouter des habitants** : Choisir l'option "Add Inhabitants" pour ajouter des habitants à un bâtiment.
- **Supprimer des habitants** : Sélectionner "Remove Inhabitants" pour retirer des habitants d'un bâtiment.
- **Ajouter des travailleurs** : Choisir "Add Workers" pour ajouter des travailleurs.
- **Supprimer des travailleurs** : Sélectionner "Remove Workers" pour retirer des travailleurs.
- **Affichage des informations d'un bâtiment** : Faire un clic gauche sur un bâtiment déjà construit pour afficher ses informations.

Des captures d'écran de notre jeu illustrent ces fonctionnalités.

1) Lancement d'une partie



2) En cours de partie



3) Fin de partie / GameOver

3. Conclusion :

En conclusion, ce projet nous a permis d'aborder les défis complexes de la création d'un jeu de stratégie et de prendre des décisions importantes concernant l'intégration des différents aspects du jeu.

L'utilisation du **Domain-Driven Design (DDD)** a été un atout majeur, nous permettant de travailler de manière structurée en divisant le projet en sous-domaines, ce qui a grandement contribué à la cohérence du tout. Les *Value Objects*, *Entities*, et *Aggregates* définis par le DDD ont joué un rôle clé pour représenter efficacement les concepts fondamentaux du jeu, tels que les types de bâtiments et les ressources.

En plus de la conception, l'ajout d'une interface graphique a grandement amélioré l'expérience utilisateur, rendant notre jeu plus agréable à jouer. Nous avons veillé à ce que l'ajout et la modification des bâtiments soient simples et intuitifs.

Au final, ce projet nous a permis de renforcer nos compétences en programmation orientée objet et d'approfondir notre réflexion sur la conception d’un jeu. Bien que le jeu soit relativement simple en termes d'options, nous estimons avoir atteint les objectifs principaux de notre projet.

Fín.