

Projet Programmation avancée : Création d'un jeu de simulateur de potager

Rapport

Année : 2024/2025

Introduction	4
Univers du jeu et ses spécificités	5
Un monde vivant et dynamique	5
Météo et conditions naturelles	6
Menaces naturelles : animaux et maladies	6
Interaction, économie et progression	6
Objectif du joueur	7
Possibilités des joueurs et déroulement d'une partie	8
Démarrage de la partie	8
Actions disponibles par tour	8
Progression temporelle	9
Interaction avec les urgences et Impact économique	9
Conditions de victoire ou de fin de partie	10
Modélisation objet	11
Diagrammes	11
Explication des classes principales	12
Tests réalisés	15
Tests fonctionnels	15
Menu principal et navigation	15
Semis de plantes	15
Croissance et arrosage	15
Récolte et inventaire	15
Vente et économie	16
Traitement et maladies	16
Tests d'événements aléatoires	16
🌤 Météo	16
🐛 Animaux nuisibles	16
🌀 Intempéries	16
Tests de résilience et de sécurité	16
Limites des tests	17
Gestion de projet	18
Organisation et communication	18
Bilan critique	20
Points forts du projet	20
Bonne structuration du code (POO)	20
Jeu fonctionnel et complet	20
Aspect ludique et pédagogique	20
Difficultés rencontrées	20
Gestion de l'affichage console	20
Complexité des interactions croisée	21
Absence de tests automatisés	21
Limites de la version actuelle	21

Pistes d'amélioration	21
Interface graphique	21
Système de sauvegarde/chargement	21
IA, scénarios et progression	22
Tests unitaires	22
Conclusion	23

Introduction

Ce projet nous a été présenté dans le cadre de notre cours de programmation orientée objet (POO) en langage C#. Il avait pour objectif de concevoir et de développer un jeu en mode Console, en appliquant les principes fondamentaux de la programmation orientée objet. Le rendu final du projet était attendu après plusieurs semaines de travail, mêlant cours encadrés et travail autonome.

Notre groupe était composé de deux personnes, et nous avons décidé de nous organiser de manière collaborative. Après plusieurs séances de réflexion et de recherche communes afin de définir l'architecture globale du jeu, nous avons réparti les tâches de développement de façon équilibrée. Chaque membre s'est chargé d'implémenter des fonctionnalités spécifiques, tout en maintenant une cohérence générale dans le code et une communication régulière.

Le jeu que nous avons conçu s'intitule **ENSemec**. Il s'agit d'un jeu de gestion agricole dans lequel le joueur possède un terrain et peut y semer, arroser, entretenir et récolter différentes plantes. Le but du jeu est d'optimiser la croissance de ses cultures en tenant compte des cycles de vie des plantes, des ressources disponibles et des événements pouvant influencer le rendement. Ce projet nous a permis de mettre en pratique nos connaissances en C#, tout en relevant un défi de conception complet et motivant

Univers du jeu et ses spécificités

Le jeu ENSemec propose un univers de simulation agricole riche et interactif, ancré dans une représentation semi-réaliste de la nature et du temps. Le joueur incarne un jardinier-gestionnaire qui débute avec un petit terrain et un budget initial. Son objectif est de faire prospérer son jardin en semant des plantes, en les entretenant et en les récoltant, tout en gérant les nombreux aléas de l'environnement. Le jeu invite ainsi le joueur à développer une stratégie adaptative et à prendre des décisions à court et long terme pour maintenir un écosystème sain et productif.

Un monde vivant et dynamique

L'environnement du jeu est régi par un **cycle saisonnier** complet : chaque saison (printemps, été, automne, hiver) dure douze semaines et impacte directement la croissance et la santé des plantes. Par exemple, l'été est favorable à la croissance rapide grâce à l'ensoleillement, mais comporte aussi des risques de sécheresse. En hiver, les températures basses ralentissent voire arrêtent la croissance, forçant le joueur à adapter son choix de culture.

Le terrain du joueur doit être personnalisé au début de la partie selon trois types de sols :

- **Terre** : sol neutre et polyvalent, idéal pour la majorité des cultures.
- **Sable** : bien drainé mais pauvre en rétention d'eau, il convient mieux à certaines plantes robustes.
- **Argile** : très riche en nutriments mais retient facilement l'eau, ce qui peut nuire aux plantes sensibles.

Il est important de préciser que dans notre implémentation actuelle, ces types de terrain sont distingués uniquement par leurs classes respectives (Sable, Argile, Terre) qui héritent de la classe de base **Terrain**. La seule différence réside dans la valeur passée au constructeur **public Terrain(string type, double surface)** : chaque classe dérivée transmet simplement un type différent ("Sable", "Argile" ou "Terre"). Ces classes ne redéfinissent aucune méthode et ne possèdent aucun comportement ou propriété propre

supplémentaire. Il n'y a donc pas encore de différenciation fonctionnelle entre les types de terrain au niveau du gameplay. Cette structure a toutefois été prévue pour permettre des évolutions futures où chaque type de sol pourrait influencer plus finement la croissance, la rétention d'eau ou la propagation des maladies.

Météo et conditions naturelles

Chaque semaine de jeu introduit une nouvelle **météo** aléatoire incluant :

- Une température moyenne (en °C)
- Un taux de pluie (0 à 1)
- Un taux d'ensoleillement (0 à 1)

Ces données influencent directement le comportement des plantes. Des conditions favorables peuvent accélérer la croissance, tandis que des conditions extrêmes (grêle, tempête, canicule) peuvent causer des dégâts, voire détruire une partie des cultures si aucune mesure de protection n'est prise.

Menaces naturelles : animaux et maladies

Des **animaux nuisibles** peuvent apparaître aléatoirement : lapins, pigeons, taupes ou chenilles. Chacun possède une probabilité d'apparition, un schéma de déplacement et un pouvoir de nuisance spécifique. Ils peuvent se déplacer vers les plantes et leur infliger des dégâts si elles ne sont pas protégées.

Par ailleurs, des **maladies** telles que le mildiou, la rouille ou la cochenille peuvent infecter les plantes. Chaque maladie est caractérisée par un taux d'apparition, une durée d'infection et une gravité. Le joueur doit surveiller attentivement l'état de ses cultures et appliquer des traitements lorsqu'une plante tombe malade.

Interaction, économie et progression

Le jeu propose une dimension économique :

- Le joueur démarre avec un budget limité.

- Il peut acheter des plantes et des outils (arrosoirs, traitements).
- Les récoltes peuvent être vendues en fonction de leur quantité et de leur qualité (impactée par la santé de la plante, la saison et le terrain).

Ce cycle achat-entretien-récolte-vente constitue le cœur du gameplay. Il pousse le joueur à bien gérer ses ressources et à anticiper les périodes critiques.

Objectif du joueur

L'objectif principal est de maintenir un jardin florissant sur le long terme. Il s'agit d'atteindre un équilibre entre rendement agricole, gestion des ressources, réaction face aux aléas, et développement durable des cultures. Le jeu encourage la planification, l'observation, et la réactivité.

En somme, ENSemec est bien plus qu'une simple simulation agricole : c'est une expérience immersive qui conjugue gestion, stratégie, adaptation et plaisir de voir son jardin évoluer au fil des saisons.

Possibilités des joueurs et déroulement d'une partie

Le jeu ENSemec place le joueur dans un rôle actif de gestionnaire agricole, où il doit prendre des décisions stratégiques à chaque tour pour optimiser la santé et la productivité de son jardin. Le déroulement d'une partie repose sur un système de tours hebdomadaires simulant le passage du temps dans l'univers du jeu.







Démarrage de la partie





Au début de la partie, le joueur :

- choisit un type de terrain (Terre, Sable, Argile),
- reçoit un budget initial, un petit stock d'outils (arrosoirs, traitements).

Actions disponibles par tour

Le menu du joueur lui propose les actions suivantes, chacune représentée par une icône pour améliorer l'expérience utilisateur :

-  **Semer** : permet de planter une plante dans une parcelle libre. Le joueur peut choisir parmi les plantes disponibles dans un petit catalogue s'il a des semences gratuites, ou acheter des plantes dans le magasin.
-  **Arroser** : essentiel pour favoriser la croissance des plantes. Chaque plante a besoin d'un arrosage régulier.
-  **Traiter** : appliquer un traitement pour soigner une infection.
-  **Désherber** : supprime les mauvaises herbes présentes sur le terrain, ce qui peut améliorer la croissance ou éviter une infection.
-  **Magasin** : ouvre l'interface de la boutique où le joueur peut acheter des plantes, des outils (arrosoirs, traitements) ou vendre ses récoltes.
-  **Passer au tour** : termine le tour actuel et déclenche la progression du temps, avec mise à jour des états du jardin (croissance, météo, maladies, événements).

-  **Afficher les recommandations** : donne des conseils en fonction de la saison et du type de plante, afin d'optimiser les décisions du joueur.
-  **Afficher jardin** : permet de visualiser graphiquement (en version console) la disposition des plantes sur le terrain.
-  **Afficher les états des plantes** : liste les plantes avec leurs caractéristiques actuelles (santé, taille, âge, état de croissance).
-  **Quitter** : permet d'arrêter la partie à tout moment.

À chaque tour, le joueur dispose d'un nombre limité d'actions (3 par semaine). Il doit donc faire des choix judicieux

Progression temporelle

Chaque action du joueur est suivie par une évolution automatique de l'environnement :

- La météo de la semaine est générée et impacte directement les plantes.
- Les maladies et les animaux nuisibles peuvent apparaître.
- Les plantes grandissent ou dépérissent selon leur environnement, leur santé et leur entretien.

Le cycle des saisons progresse automatiquement, influençant la productivité des plantes. Certaines plantes poussent mieux à certaines saisons, ce qui encourage le joueur à planifier ses semis en conséquence.

Interaction avec les urgences et Impact économique

Des événements imprévus comme des tempêtes, des sécheresses ou des attaques d'animaux peuvent survenir. Lorsqu'une urgence est déclenchée, le joueur est contraint de la gérer avant de pouvoir reprendre le cours normal du jeu. Ces situations ajoutent de la tension stratégique et forcent à adapter sa planification.

Le joueur peut vendre ses récoltes pour gagner de l'argent, ce qui lui permet d'acheter de nouvelles plantes ou des outils. La qualité des produits récoltés dépend de la

santé et de la croissance des plantes, influencée par l'entretien et les conditions météo. Le jeu propose donc une boucle économique complète, récompensant les décisions optimisées.

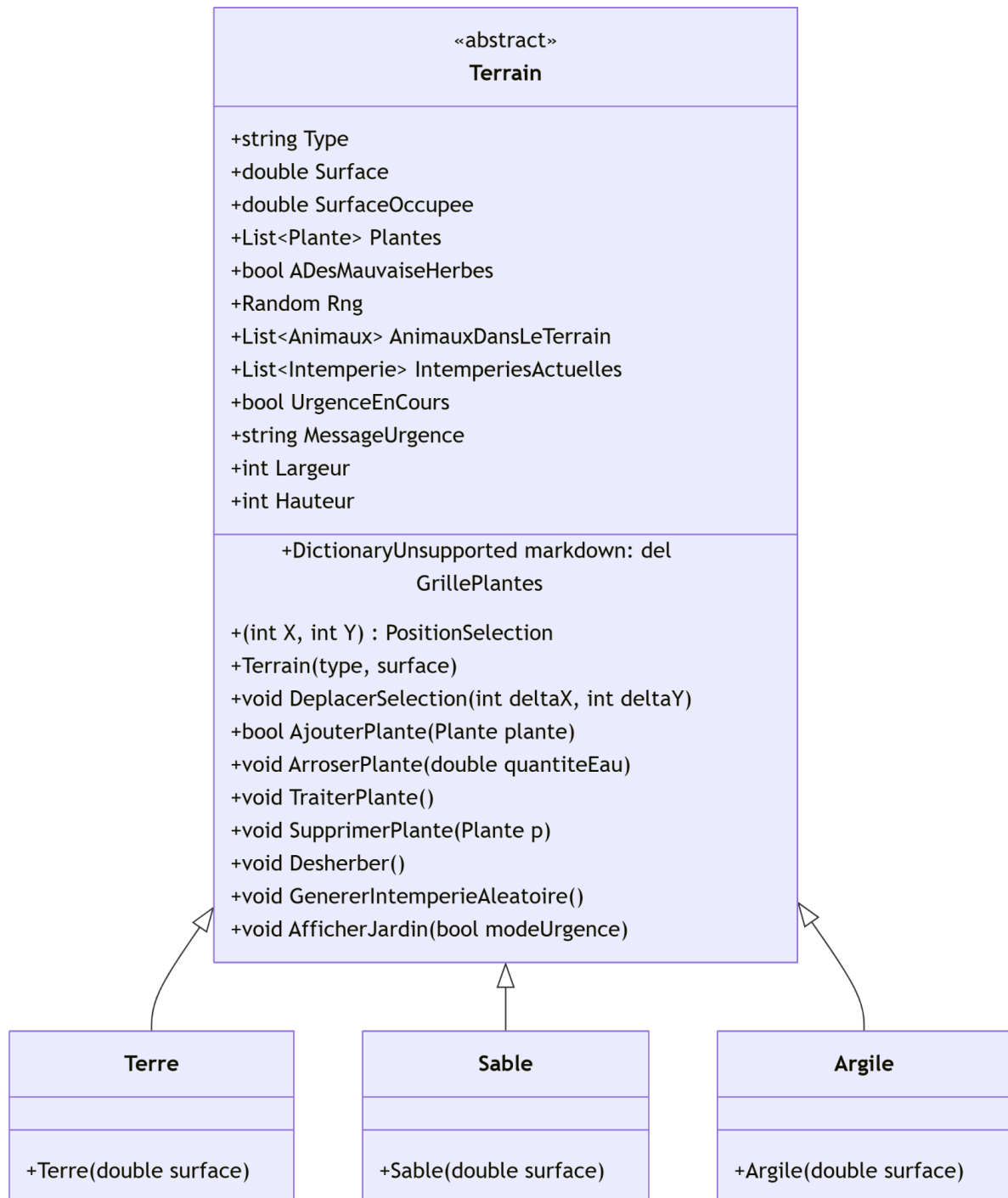
Conditions de victoire ou de fin de partie

La partie se termine dans l'un des cas suivants :

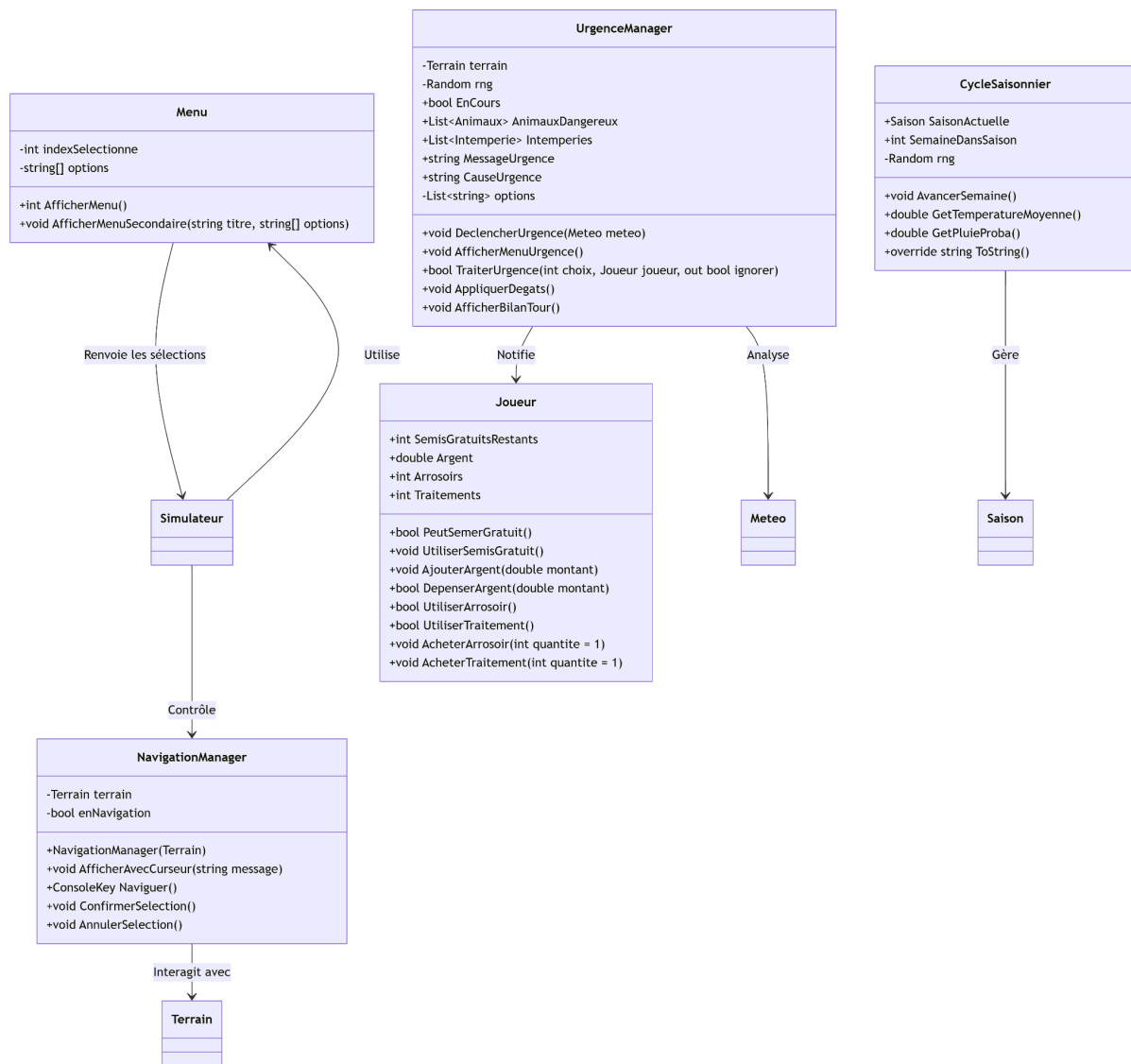
- Le joueur choisit de quitter le jeu.
- Le joueur n'a plus d'argent, plus de semis gratuits, et aucune récolte à vendre, ce qui l'empêche d'effectuer toute nouvelle action. Le jeu détecte alors qu'il n'est plus possible de progresser, et la partie s'arrête automatiquement

Modélisation objet

Diagrammes



Hiérarchie des Terrains



Système de gestion

Deux autres diagrammes — hiérarchie des plantes et diagramme UML complet— n'ont pas été insérés ici en raison de problèmes de lisibilité liés à la qualité d'affichage des images dans Google Docs.

Vous pouvez les consulter en haute qualité via le lien suivant :

<https://drive.google.com/drive/folders/1K0Q34FWdLOM-hZs9fmjRHdE5-6sJq6o7?usp=sharing>

Explication des classes principales

Plante (abstraite) : Classe de base pour toutes les plantes du jeu. Contient les attributs communs (santé, taille, besoins en eau, maladies, etc.) et les méthodes d'entretien (arrosage, traitement, etc.).

Comestible / Fleur / Medicinale : Sous-classes spécialisées de Plante, représentant différents types de cultures.

Terrain (abstraite) : Représente une parcelle cultivable. Contient les plantes, leur position, les mauvaises herbes, les intempéries en cours, et permet de gérer les interactions (ajouter, arroser, supprimer une plante).

Terre / Sable / Argile : Sous-classes concrètes de Terrain, représentant différents types de sol.

Joueur : Stocke les ressources du joueur : argent, semis gratuits, outils (arrosoirs, traitements). Gère les achats, les dépenses, l'utilisation des outils.

Simulateur : Classe centrale du jeu. Orchestration de toutes les autres classes (terrain, météo, joueur, urgences, etc.). Gère les actions par tour, l'évolution du temps, et la progression globale du jeu.

CycleSaisonnier : Gère l'alternance des saisons et l'avancement hebdomadaire. Fournit la saison actuelle et les probabilités climatiques (pluie, température, etc.).

Meteo : Contient les conditions météo : ensoleillement, pluie, température, et si le climat est extrême. Impacte directement la croissance des plantes.

Maladie : Représente une maladie que peut attraper une plante. Décrit le risque, la durée, la probabilité d'apparition et les effets.

Animaux : Entités aléatoires pouvant attaquer les plantes. Ils se déplacent sur le terrain et infligent des dégâts s'ils ne sont pas arrêtés.

Intemperie : Événements climatiques violents (grêle, tempête, etc.) affectant temporairement les plantes.

UrgenceManager : Gère les événements imprévus (animaux, intempéries). Affiche un menu spécifique et bloque le jeu jusqu'à résolution du problème.

Inventaire : Stocke les récoltes après chaque tour. Permet de les vendre au magasin. Sert de liaison entre la production et l'économie du joueur.

Recolte : Objet représentant une récolte individuelle : nom, type de produit, quantité, qualité, saison de récolte. Utilisé dans l'inventaire et le calcul du prix de vente.

Magasin : Interface marchande du jeu. Permet au joueur d'acheter des plantes et outils, ou de vendre ses récoltes. Gère les prix dynamiques.

Menu : Gère les menus textuels affichés dans la console. Affiche les options disponibles et capture la sélection du joueur.

NavigationManager : Permet au joueur de déplacer un curseur sur le terrain pour choisir une plante ou une case. Gère l'interface interactive du jardin.

Tests réalisés

Pour garantir le bon fonctionnement de l'application ENSemec, plusieurs tests ont été réalisés tout au long du développement. Ces tests ont porté sur la logique métier, l'affichage en console, les interactions utilisateur et la gestion des événements aléatoires. Ils ont été effectués de manière manuelle à chaque itération, car l'application repose principalement sur une interface console interactive.

Tests fonctionnels

Menu principal et navigation

- Vérification du bon affichage du menu principal.
- Test de la navigation via les touches fléchées et de la sélection des options (semis, magasin, etc.).
- Gestion correcte du retour au menu après chaque action.

Semis de plantes

- Semis de toutes les catégories de plantes (comestibles, médicinales, fleurs).
- Test de l'utilisation de semis gratuits et achat via le magasin.
- Blocage correct en cas de surface insuffisante ou de case déjà occupée.

Croissance et arrosage

- Observation de la croissance des plantes sur plusieurs tours.
- Effets visibles d'un arrosage insuffisant ou excessif sur la santé et la taille.
- Impact des saisons et de la météo sur la croissance.

Récolte et inventaire

- Vérification que les plantes à maturité deviennent récoltables.
- Ajout correct des récoltes à l'inventaire.
- Qualité et quantité des produits bien influencées par les soins et la météo.

Vente et économie

- Test de la vente de récoltes avec calcul du prix basé sur la qualité.
- Augmentation de l'argent du joueur après chaque vente.
- Achat d'outils (arrosoirs, traitements) fonctionnel via le magasin.

Traitement et maladies

- Application correcte des traitements (si plante malade).
- Vérification de la disparition de la maladie et de la régénération de santé.
- Comportement correct si la plante est déjà saine.

Tests d'événements aléatoires



Météo

- Génération de météo réaliste selon la saison.
- Variation des températures, pluie et ensoleillement impactant les plantes.
- Affichage correct et coloré des conditions météo.



Animaux nuisibles

- Déplacement aléatoire des animaux sur le terrain.
- Attaque de plantes non protégées, application de dégâts.
- Repousser les attaques avec les protections physiques ou logiques.



Intempéries

- Apparition d'intempéries (grêle, tempête, sécheresse).
- Effets visibles sur les plantes (dégâts, mort).
- Gestion des urgences (menu bloquant tant que l'urgence n'est pas traitée).

Tests de résilience et de sécurité

- Test de comportement en cas de fonds insuffisants.

- Blocage de l'accès aux actions impossibles (pas d'arrosoirs, plus de semis).
- Gestion des entrées utilisateur invalides dans les menus.
- Vérification de la fin automatique de partie si le joueur n'a plus de ressources ni de possibilités d'action.

Limites des tests

Actuellement, aucun test automatisé unitaire (ex. avec NUnit ou MSTest en C#) n'a été mis en place. Tous les tests ont été réalisés manuellement via des scénarios d'usage. L'interface orientée console rend l'automatisation complexe, mais une refonte du moteur en architecture MVC pourrait permettre des tests unitaires à l'avenir.

Gestion de projet

Organisation et communication

Concernant l'organisation du projet, nous avons utilisé GitHub afin de collaborer efficacement en parallèle sur le code. Cet outil nous a permis de suivre l'évolution du projet et de travailler simultanément, bien que nous ayons rencontré certaines difficultés pour résoudre les conflits de fusion.

Pour la communication, nous avons principalement échangé via WhatsApp ainsi que pendant les séances de travail en présentiel, ce qui nous a également permis de poser nos questions directement aux enseignants concernés.

Tâches	Description	Temps estimé
Découverte du sujet	Analyser attentivement l'ensemble du sujet afin d'identifier les éléments essentiels à implémenter	1h
Règle de jeu	comprendre les règles (comment gagner- perdre)	1h
Définitions des classes	Définir les classes dont on a besoin	1h
Création d'un diagramme des classes	Création un diagramme en utilisant Canva	1h
Dessiner l'affichage souhaitée	Sur une feuille on a dessiné comment on veut obtenir l'affichage à la fin du jeu	30 minutes
Coder toutes les classes	Création des classes , codage	30h
Test d'intégration	Vérification du bon fonctionnement de	3h

	l'enchaînement des classes et des méthodes	
Rapport	Rédaction pendant toute la durée du projet.	4h

Planification détaillé du projet

Bilan critique

Points forts du projet

Bonne structuration du code (POO)

Le projet s'appuie sur une modélisation orientée objet solide. Le découpage en classes cohérentes (plantes, terrain, météo, joueur, simulateur, etc.) facilite la lisibilité, la maintenance et les évolutions futures.

Jeu fonctionnel et complet

L'ensemble des fonctionnalités prévues a été intégré :

- gestion des semis, croissance, récolte ;
- météo dynamique ;
- magasin avec achats/ventes ;
- urgences aléatoires (animaux, intempéries) ;
- système d'économie interne.

Le jeu peut être joué du début à la fin sans erreur bloquante, ce qui démontre une bonne robustesse globale.

Aspect ludique et pédagogique

Malgré une interface en console, l'utilisation d'icônes (🌱, 💧, 🐷...) et d'éléments dynamiques (couleurs, météo, urgences) rend l'expérience engageante. Le jeu sensibilise aussi aux facteurs environnementaux influant sur l'agriculture.

Difficultés rencontrées

Gestion de l'affichage console

Mettre en œuvre une interface utilisateur ergonomique avec uniquement la console a nécessité de nombreuses astuces (position du curseur, couleurs, menus dynamiques). Cela a parfois ralenti le développement.

Complexité des interactions croisée

Certaines mécaniques, comme l'interaction entre météo, maladies, terrain et croissance, ont nécessité une coordination précise entre plusieurs classes, augmentant la complexité et le risque de bugs.

Absence de tests automatisés

Par manque de temps, nous n'avons pas mis en place de tests unitaires. Toute la validation s'est faite par tests manuels, ce qui peut engendrer des régressions en cas de modification future du code.

Limites de la version actuelle

- Interface uniquement textuelle (console).
- Pas de persistance des données (pas de sauvegarde/chargement).
- Aucune musique ou effet sonore.
- Les animaux et les intempéries sont encore relativement simples et peu stratégiques.
- L'IA ou comportement autonome est minimal (pas de compétition, de classement, etc.).

Pistes d'amélioration

Interface graphique

Passer à une interface graphique (via Windows Forms, WPF ou Unity) permettrait une expérience beaucoup plus immersive et intuitive, en particulier pour la gestion du terrain et des événements.

Système de sauvegarde/chargement

Ajouter une persistance des données (JSON, XML ou base de données locale) pour permettre au joueur de reprendre sa partie à tout moment.

IA, scénarios et progression

Ajouter des objectifs, missions, ou niveaux de difficulté. Implémenter une IA concurrente (autre agriculteur virtuel) enrichirait l'aspect stratégique.

Tests unitaires

Structurer le code métier en couches (modèle MVC par exemple) pour permettre la mise en place de tests automatisés et garantir la stabilité dans le temps.

Le projet ENSemec a rempli son objectif initial : créer un simulateur agricole jouable, réaliste et évolutif. Bien que des limites existent (liées à l'interface et à l'absence de tests automatisés), la base du jeu est solide, extensible et offre un excellent point de départ pour une version future plus ambitieuse. Ce projet nous a permis de consolider nos compétences en programmation orientée objet, gestion d'événements, interface utilisateur console, et conception logicielle.

Conclusion

Le développement du jeu ENSemec nous a permis de mener à bien un projet complet mêlant modélisation objet, gestion d'un système complexe et interaction utilisateur. À travers ce simulateur agricole, nous avons conçu un environnement riche, dynamique et évolutif, capable de refléter les réalités d'une gestion de jardin confrontée à des paramètres climatiques, biologiques et économiques.

Au-delà des compétences techniques en programmation orientée objet, ce projet nous a permis d'expérimenter la conception modulaire, la gestion d'événements aléatoires, l'affichage interactif en console, et la prise de décision stratégique par le joueur.

Même si certaines fonctionnalités restent perfectibles (interface graphique, tests automatisés, IA), la version actuelle constitue une base robuste, stable et riche en potentiel. Elle offre une expérience ludique, pédagogique et engageante.

Enfin, ce projet a été aussi une expérience humaine précieuse, basée sur la collaboration, l'adaptabilité et la résolution de problèmes en équipe — des qualités indispensables dans tout développement logiciel.