DERAMAIX Mathilde

Hout Dounia

Groupe 2

Projet de POO

Introduction

Ce document de justifications techniques a pour objectif de clarifier les choix de conception et les mécanismes mis en œuvre dans le développement du simulateur de potager ENSemenC. Ce jeu propose au joueur de cultiver un jardin dans un environnement choisi pouvant être réel ou imaginaire, influençant les types de semis disponibles et les conditions climatiques rencontrées. L'objectif est de maintenir en vie, faire prospérer et valoriser différentes plantes, tout en gérant les aléas de la nature et de l'environnement.

Le jeu repose sur deux modes distincts : un mode classique, simulant l'évolution du potager semaine après semaine, et un mode urgence, déclenché aléatoirement en cas d'intempéries ou d'intrusion. Le joueur dispose alors d'un ensemble d'actions à réaliser pour assurer la survie de ses cultures.

Ce projet a été développé sous Visual Studio, en C# avec le framework .NET 8, et hébergé sur GitHub afin de faciliter le travail collaboratif en binôme. L'ensemble du code respecte les principes de la programmation orientée objet, en intégrant notamment l'héritage, les classes abstraites et l'encapsulation.

1. Le jeu

A. Univers

Le jeu se déroule dans un monde où le joueur incarne un jardinier, responsable d'un potager situé dans un pays choisi au début de l'aventure. Ce pays, qui peut être imaginaire, influe directement sur le type de semis disponible et sur les conditions météorologiques que le joueur devra gérer. À chaque début de partie, le joueur reçoit une quantité de semis et plusieurs terrains avec des caractéristiques uniques.

Le jardin est constamment surveillé par une webcam qui peut envoyer des alertes en cas de conditions météorologiques extrêmes ou si des intrus (animaux ou parasites) apparaissent dans le jardin. Le but du jeu est de cultiver des plantes, de les faire pousser et de les récolter afin de pouvoir les vendre et d'avoir le plus d'argent possible. Cependant, si les conditions ne sont pas favorables ou que les mauvaises herbes et les parasites ne sont pas maîtrisés ou que les plantes sont malades trop longtemps, le jardin risque de se détériorer et les plantes de mourir.

B. Spécificités

Les plantes sont modélisées à partir d'une classe Plante, dont héritent différentes classes concrètes telles que PlanteSucree et PlanteAcidulee et chaque plante possède des caractéristiques spécifiques telles que

- ➔ Sa nature : annuelle, vivace, comestible, ornementale, mauvaise herbe, etc.
- ➔ Son terrain préféré : terre, sable, argile.
- ➔ Sa vitesse de croissance qui est influencée par l'environnement.
- ➔ Ses besoins en eau et en luminosité.
- ➔ Sa zone de température idéale.
- ➔ Les maladies possibles qu'elle peut contracter (avec une probabilité aléatoire).
- ➔ Son espérance de vie, et le nombre de produits récoltables par plant. (restera à modifier ce qu'on a enlevé)

Si les conditions environnementales ne sont respectées qu'à moins de 50 %, la plante meurt prématurément. Au contraire, plus les conditions sont favorables, plus elle pousse rapidement.

Les terrains sont eux aussi modélisés à l'aide d'une classe Terrain, avec des dérivées spécifiques (par exemple TerrainSucre, TerrainAcidule. Chaque terrain possède :

- ➔ Un nom
- ➔ Un nombre de plantes
- ➔ Une surface disponible
- ➔ Une capacité d'accueil de plantes
- ➔ Un nb de pièces en or en chocolat
- ➔ Une météo
- ➔ Une liste de plantes disponibles
- ➔ Un niveau d'humidité du sol
- ➔ Un type de sol
- ➔ Une longueur et une largeur
- ➔ des booléens EstRecouvertDePlantesMortes, IntepériesDetectees, IntrusDetecte qui seront utilisés dans les méthodes de la classe terrain

Le simulateur de potager propose deux modes de jeu, chacun avec ses propres mécaniques :

- ➔ Mode Classique

  Ce mode représente la gestion "courante" du potager. Chaque tour correspond à un mois écoulé, durant lequel le joueur peut planifier et réaliser plusieurs actions : semer de nouvelles graines, arroser les plantes, désherber, ou encore récolter ses plantes.

Une simulation de la météo rythme ce mode (aléatoire selon le mois) : les températures, précipitations et le niveau d'ensoleillement influencent directement la croissance et la santé des plantes.

➔ Mode Urgence

Ce mode s'active lorsqu'un événement inhabituel survient, comme l'apparition d'un intrus (rongeur, oiseau…) ou une intempérie violente (grêle, tempête…). Ces informations sont communiquées aléatoirement par la webcam qui récupère les informations météo.

Le joueur bascule alors dans une vue dynamique du potager, où il doit intervenir rapidement pour limiter les dégâts, un menu d'action d'urgence lui est alors proposé afin de sauver son potager.

C. Possibilité des joueurs

Le jeu se joue en solo : le joueur incarne un jardinier chargé de gérer et de faire prospérer son potager au fil du temps. À chaque tour, il dispose d'un ensemble d'actions possibles, qu'il doit choisir de manière stratégique en fonction des conditions météorologiques et de l'état de ses cultures. L'objectif principal est de maintenir un potager en bonne santé, d'obtenir des récoltes abondantes, et de faire face aux aléas climatiques et biologiques qui peuvent perturber la croissance des plantes. Le joueur progresse ainsi à son rythme dans un environnement vivant, à la fois stratégique et évolutif.

D. Déroulé d'une partie

Au lancement du jeu, le joueur commence par choisir le monde dans lequel il souhaite faire évoluer son potager.

Une fois le monde sélectionné, le joueur doit ensuite choisir son type de terrain : sucré ou acidulé, les semis ne sont pas les mêmes donc leur caractéristiques sont également différentes. Le jeu progresse ensuite mois après mois, à chaque début de mois, le joueur accède aux informations suivantes :

- ➔ Informations météo
- ➔ Informations du terrain qui indique le type de sol, l'humidité, le nombre de plantes présentes, la capacité du terrain et le stock de semis disponibles.
- ➔ Informations sur les conditions favorables des semis
- ➔ Informations webcam indiquant si une urgence à été détectée ou non pour le mois.
- ➔ Nombre de pièces d'or en chocolat qu'il possède s'il a choisi de vendre ses semis auparavant.

Chaque mois, le joueur peut choisir trois actions parmi les huit proposées

```
Que souhaitez-vous faire ce mois-ci ? (Vous pouvez faire 3 choix) Action : 1
1. Arroser
2. Récolter de nouvelles plantes
3. Désherber
4. Semer
5. Soigner
6. Déraciner les plantes fanées
7. Vendre des semis
8. Ne rien faire
```

Une fois les trois actions réalisées, le mois prend fin, les plantes évoluent selon la météo et l'état du potager. Avant de débuter un nouveau mois, un résumé du terrain avec l'état des plantes apparaît.

La partie peut se terminer dans plusieurs cas :

- ➔ si son potager devient inexploitable (toutes les plantes sont mortes);
- ➔ si il n'y a plus de semis en stock et que le terrain est vide;
- ➔ si le joueur souhaite simplement arrêter sa partie à tout moment (choix à la fin de chaque mois);

2. Aspect technique

1. Diagramme des classes

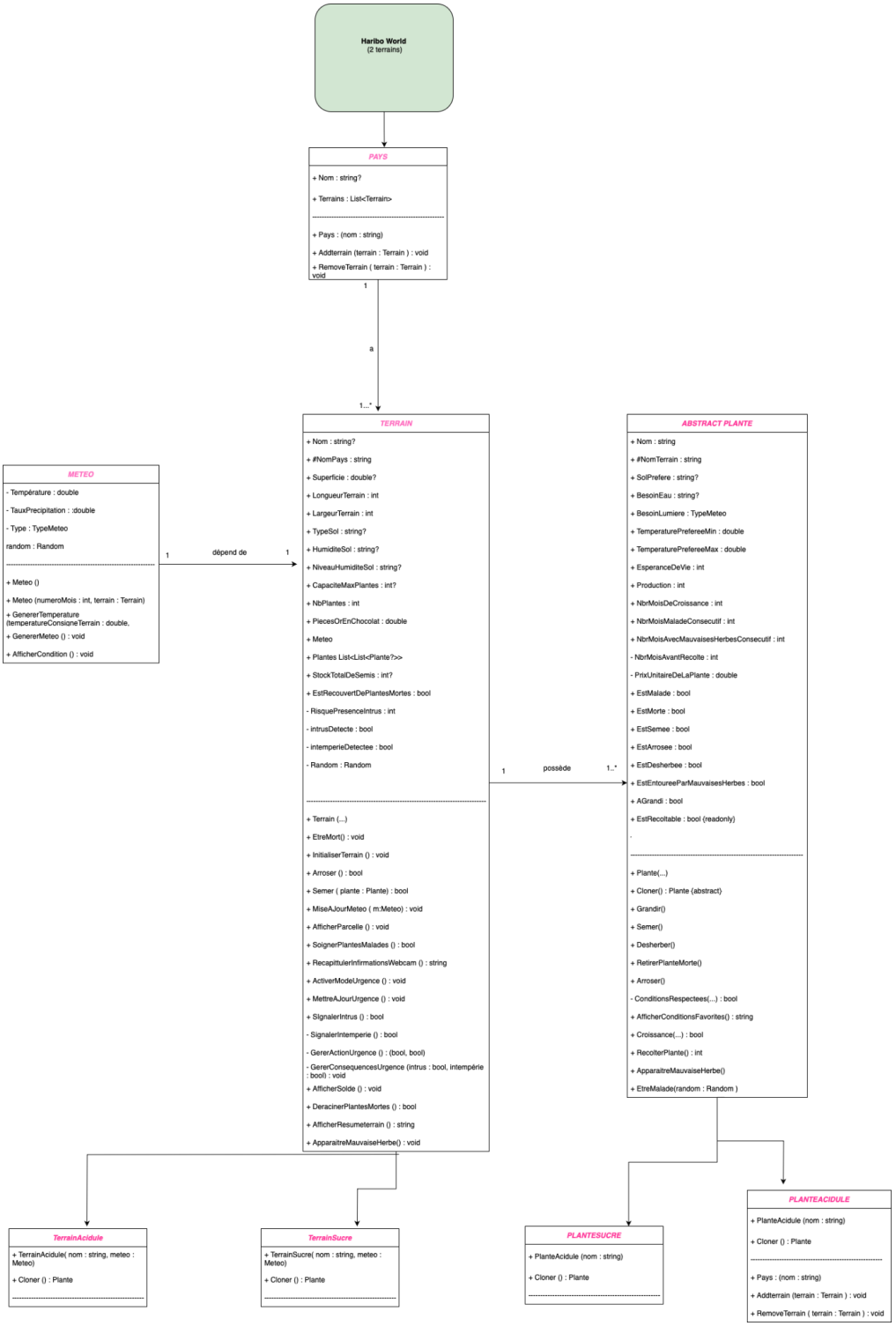

Diagramme des classes général

2. Choix techniques

Afin de structurer notre jeu en programmation orientée objet, nous avons fait plusieurs choix techniques afin d'assurer une bonne organisation du code, une gestion claire des actions, et une logique cohérente à l'échelle du terrain et des plantes.

Pour commencer, nous avons créé dans la classe Plante.cs des méthodes spécifiques dont le rôle est uniquement de mettre à jour l'état d'une plante avec des booléens. Ces méthodes centralisent toutes les modifications relatives à la croissance, à la santé ou à l'humidité d'une plante et autres. Plutôt que de gérer chaque plante individuellement depuis l'extérieur, nous avons fait le choix de réutiliser ces méthodes dans la classe Terrain.cs via une autre méthode qui applique les changements à l'ensemble des plantes présentes sur le terrain. Cette généralisation se fait au moyen de deux boucles for imbriquées, qui parcourent la grille de plantes et permettent ainsi d'appliquer de façon systématique et ordonnée les mises à jour à chaque plante.

Pour gérer la logique du jeu, notamment la fin de partie et le contrôle des saisies utilisateur, nous avons utilisé des variables booléennes associées à des boucles while. Cela nous a permis de vérifier la validité des entrées et de gérer notamment les fin de parties ou la mort de certaines plantes du potager.

En ce qui concerne la gestion des différentes options et actions disponibles, nous avons privilégié l'utilisation de switch-case. Ce choix technique s'explique par la nature définie des possibilités du jeu. Par exemple, pour déterminer l'humidité en fonction des conditions météorologiques telles que ensoleillé ou nuageux, ou encore pour gérer les actions prédéfinies du joueur, le switch-case nous a permis de traiter efficacement chaque cas, puisque même si le nombre d'actions est important, il reste connu, évitant ainsi la complexité d'une gestion infinie.

Pour faire apparaître des intrus ou d'autres événements imprévus, on utilise la propriété Random. Cela rend le jeu plus vivant et plus réaliste, tout en gardant un certain contrôle sur la fréquence de ces événements.

Enfin, nous avons fait le choix de centraliser la majorité des méthodes dans la classe Terrain. En effet, de nombreuses opérations, telles que la gestion de la météo s'appliquent naturellement à l'ensemble du terrain et non individuellement à chaque plante ou au pays.

En ce qui concerne la fonctionnalité bonus, nous avons choisi de faire un mode de vente. Le but de notre jeu est de faire pousser des semis et de les vendre afin de se faire un maximum d'argent !

3. Tests réalisés

Afin de garantir le bon fonctionnement et la stabilité de notre code, nous avons adopté une stratégie de test régulier tout au long du développement. Après chaque ajout ou modification significative, nous lancions le jeu via le terminal afin de vérifier que le comportement du programme correspondait à nos attentes. Si ce n'est pas le cas, on ajoute cette fonctionnalité à une liste d'éléments à coder pour ne pas l'oublier. On faisait la même chose si une incohérence ou une idée d'amélioration pendant les tests nous venait en tête.

Lorsque les résultats obtenus n'étaient pas conformes, nous commencions par relire attentivement le code récemment modifié pour identifier d'éventuelles erreurs de logique ou de syntaxe. Cette méthode de tests fréquents nous a permis de repérer rapidement les erreurs avant qu'elles ne s'accumulent. Le test régulier du jeu et l'utilisation de la POO, nous a également permis de débuter très tôt l'interface d'affichage ce qui nous a aidé pour mieux appréhender le déroulé du jeu.

Enfin, une fois l'application finalisée, nous avons joué plusieurs parties pour vérifier la cohérence globale et l'affichage correct du jeu et nous assurer qu'aucune anomalie ne perturbait l'expérience de jeu. Cela nous a également permis de tester différents scénarios (conditions météorologiques extrêmes, mauvaises herbes, maladies, différents terrains, etc.).

3. Gestion de projet

Pour notre projet, nous avons commencé par concevoir ensemble notre diagramme de classes et à discuter des spécificités de notre simulateur de potager. Cela nous a permis d'avoir une vision claire de l'architecture globale du code avant de nous répartir les tâches. Par exemple, l'une d'entre nous s'est concentrée sur la gestion des terrains, des semis et des événements météorologiques, tandis que l'autre a travaillé sur l'interface console, les actions du joueur et la mécanique de jeu mensuelle. Au fur et à mesure, nous avons tenu une liste de bord pour savoir ce qu'on avait réalisé ou pas encore.

Les séances de TD ont été principalement utilisées pour faire le point sur notre avancement, définir les objectifs à venir, et demander de l'aide aux chargés de TD en cas de blocage technique. Pour certaines parties plus complexes, notamment la simulation des effets de la météo sur les plantes ou la gestion des événements d'urgence, nous avons pris le temps de réfléchir ensemble sur papier à un modèle algorithmique avant de le coder.

Une fois la phase de codage terminée, nous avons mis à jour le diagramme de classes que nous avions conçu au départ, rajouter certains éléments qui n'étaient pas prévus au départ

par exemple. Nous avons également consacré du temps à commenter le code et à finaliser proprement le diagramme, afin d'assurer une bonne lisibilité et une cohérence globale. Là encore, la répartition des tâches s'est faite de manière équilibrée, chacun contribuant selon ses points forts. Notre méthode de travail a été de travailler ensemble sur les aspects critiques (comme la structure du programme ou les interactions complexes entre objets) tout en avançant individuellement sur les parties plus simples ou plus techniques. Cela nous a permis d'être à la fois productifs et rigoureux, tout en conservant une bonne dynamique de groupe.

Planning suivi:

Tâches	03/04/25	14/04/25	25/04/25	26/04/25	01/05/25 - 05/05/25	09/05/25	12/05/25 - 16/05/25	
Création du jeu								
Création du `.gitignore`								
Création des classes UML								
Dépot des classes sur github								
Définition des classes mères : Terrain, Plante,Meteo								
Creation de classes filles pour Terrain et Plante								
Interface de début de partie								
Développement des classes / méthodes de pays								
Création de la méthode webcam								
Ajout des méthodes avec probabilité								
Interface de jeu / fin de partie								
Réorganisation du code								
Refactorisation des classes (en cascade)								
Finalisation de la classe météo et ajout de méthodes actions pour le mois								
Ajout du mode urgence et gestion des bugs								
Modification de certaines méthodes de la classe plante et terrain								

4. Bilan critique du projet et conclusion

A travers ce projet, nous avons pu consolider nos compétences en programmation orientée objet c#, en mobilisant des concepts clés tels que l'héritage, les classes abstraites ainsi que la gestion des interactions complexes entre objets. Ce projet nous a également permis de progresser dans notre capacité à structurer un code sur le long terme, à détecter et corriger les erreurs, et à adopter une démarche rigoureuse de test.

Avec davantage de temps, nous aurions aimé enrichir le jeu en intégrant un nouveau monde avec un nouveau type de semi ou ajouter d'autres résultats. Malgré cela, nous sommes très satisfaites du résultat final.

En conclusion, ce projet a été une expérience très formatrice, à la fois sur le plan technique et méthodologique. Il nous a permis de mettre en pratique les principes de la programmation orientée objet dans un cadre ludique et motivant, tout en développant des compétences en gestion de projet, en autonomie, et en résolution de problèmes.