BORDEAUX INP Ensc

Projet de Programmation avancée
ENSemenC

Rapport technique

Cauzid Hugo
Valvin Émilie

1A - 23/05/2025

Table des matières

I. Introduction

Dans le cadre du cours de Programmation avancée, nous avons dû réaliser un projet en C#. Nous devions coder un jeu qui simule un potager, dans lequel le joueur peut planter, arroser ou encore cueillir des plantes. Ce jeu est matérialisé dans la console par un plateau de jeu représentant les terrains ainsi qu'une interface avec l'inventaire du joueur. Nous devions également utiliser GitHub pour coder de manière collaborative.

Ce document de justifications techniques a pour objectif de clarifier le code que nous avons produit. Pour ce faire, nous allons aborder l'univers du jeu, le déroulement d'une partie avec la description du mode classique et du mode urgence. Puis nous aborderons les différents tests effectués tout au long du projet. Nous nous attarderons ensuite sur la modélisation objet que nous avons réalisée. Finalement, nous décrirons comment nous avons géré ce projet avant de terminer par un bilan.

II. L'univers du jeu

Pour ne pas être contraints par la réalité, nous avons choisi de ne pas mettre dans notre jeu des plantes existantes. Nous ne voulions pas recréer un jeu qui existait déjà et nous ne pensions pas avoir le temps de créer entièrement de nouvelles plantes. Nous avons donc choisi des plantes issus de différents univers, notamment Harry Potter et Stardew Valley. Nous avons ajouté des éléments que nous avons imaginé nous même, comme les ratiboiseurs.

Nous avons donc établi une liste de 5 plantes : la branchiflore, le filet du diable, le fruit étoilé, la mandragore et la rose de fée. Nous avons représenté chacune de ces plantes avec un emoji dans l'inventaire et avec un dessin plus détaillé lorsque l'on demande ses informations. Nous avons également établi 4 types de terrains avec chacun des conditions différentes. Un terrain peut donc être de type dune, de type herbeux, de type jungle ou de type rocheux.

III. Le déroulement d'une partie

A - Mode classique

Lorsque le joueur lance une partie, la date, la météo et la saison s'affiche. Puis en dessous, un plateau de 4 cases s'affiche. Le joueur est sur une case et il peut se déplacer sur les cases avec les flèches du clavier. Une fois que le joueur a choisi la case, il appuie sur entrer et une liste d'actions s'affichent. Les actions que le joueur peut faire s'affichent en blanc et celles qu'il ne peut pas faire dans les circonstances actuelles sont en gris. Le joueur peut de nouveau naviguer parmi ses options avec les flèches, puis une fois qu'il a choisi l'action il clique sur entrer.

Lorsqu'il commence à jouer le joueur peut :

- Ajouter un terrain
- Laisser les plantes pousser
- Revenir en arrière
- Quitter le jeu

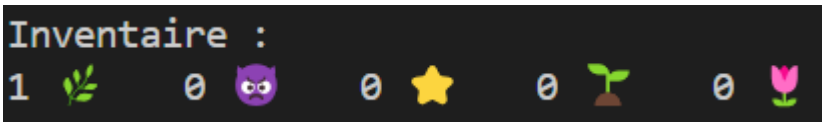
À chaque étape, on demande au joueur de valider son choix.

- Ajouter un terrain :

Le joueur choisit le type de terrain qu'il veut rajouter parmi une dune, un terrain avec de l'herbe, une jungle ou un terrain avec des roches. Il navigue parmi ces options avec les flèches, puis une fois qu'il a choisi il appuie sur entrer.

- Laisser les plantes pousser

Après avoir confirmé qu'il voulait laisser passer le temps, le joueur choisit combien de jours passent. Puis l'inventaire s'affiche avec le nombre de plantes présentes sur le plateau et leur type.



Affichage de l'inventaire après avoir laisser les plantes pousser

- Revenir en arrière

Cette action permet au joueur de retourner au mode de déplacement à travers le plateau.

- Quitter le jeu

En choisissant cette action, le joueur quitte le jeu.

Puis une fois que le joueur à ajouter un terrain de nouvelles possibilités s'offrent à lui :

- Arroser le sol
- Ajouter une plante
- Afficher les informations du terrain

- Arroser le sol

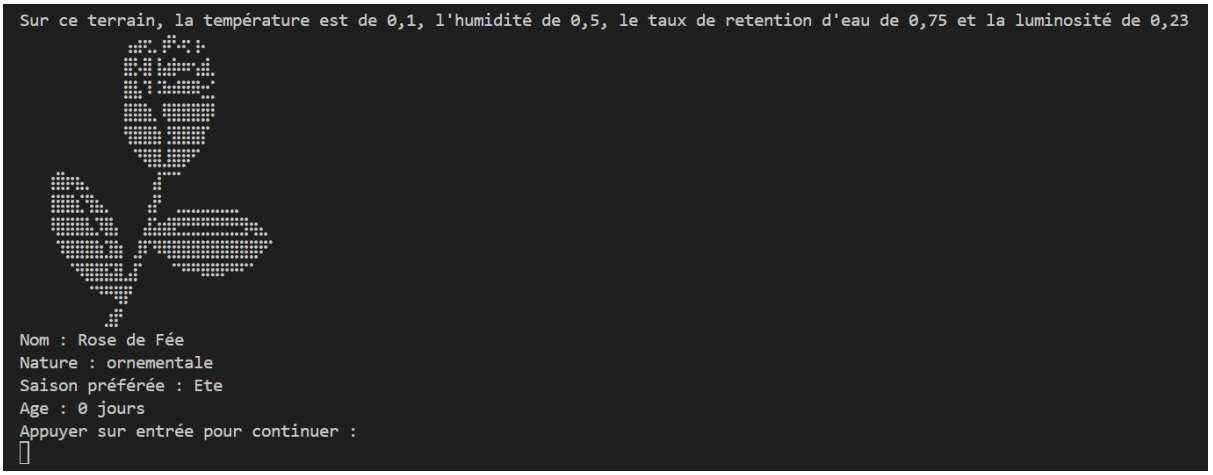
Cette action humidifie le sol pour permettre à certaines plantes de mieux pousser.

- Ajouter une plante

Une fois que le joueur a sélectionné cette action, la liste des plantes disponibles s'affiche. Le joueur en choisit une et confirme son choix, ou alors retourne en arrière.

- Afficher les informations sur le terrain

La température, l'humidité, le taux de rétention d'eau et la luminosité du terrain s'affichent. Si une plante est sur le terrain, en plus des informations sur le terrain s'affichent les informations sur la plante ainsi que sa représentation graphique.



Exemple de l'affichage de la Rose de Fée placée sur un terrain de type Jungle

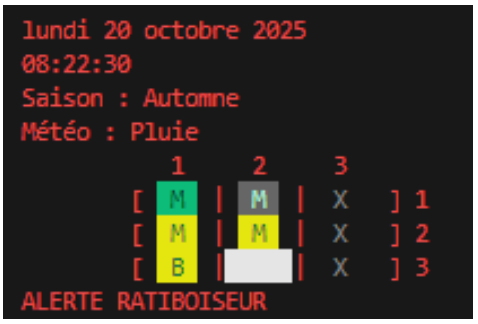
- Retirer les plantes

Une fois que le joueur a planté une plante, cette nouvelle action apparaît. Le joueur peut choisir à tout moment de retirer sa plante pour libérer de l'espace. Lorsqu'une plante meurt, le joueur doit l'enlever.

Lorsque le joueur a mis des terrains sur les 4 cases disponibles, de nouvelles cases vides apparaissent pour que le joueur puisse continuer de jouer sans avoir à retirer des plantes. Le plateau de jeu sera toujours d'une forme carrée et aura une taille maximale de 10 cases de côté.

B - Mode urgence

Le mode urgence est un mode qui se déclenche si certaines conditions sont remplies.



Exemple d'affichage du mode urgence

- Filet du diable

Le filet du diable est une plante invasive qui peut apparaître sur un terrain inoccupé. Chaque jour passé, un terrain vide a une chance sur 20 de recevoir un filet du diable. Ces plantes ne servent à rien si ce n'est occuper de l'espace de manière impromptue. Pour s'en débarrasser, il faut l'enlever de la case. Cette plante ne fait pas apparaître le mode urgence, mais c'est un élément néfaste.

- Ratiboiseurs

Les ratiboiseurs sont des animaux venant dévorer les plantes du jardin. Lorsque le jardin possède 6 plantes ou plus, il y a une chance sur 33 pour qu'un ratiboiseur arrive. Le jeu passe alors en mode urgence, vous devez trouver l'animal et le faire fuir hors du jardin avant qu'il ne mange toutes vos plantes. Cependant, un ratiboiseur est très furtif et n'est pas facilement visible, il faut examiner les cases de votre jardin pour trouver son emplacement. Lors d'une chasse au ratiboiseur, aucune autre action n'est effectuable.

IV. Les tests réalisés

Pour nous assurer du bon fonctionnement de notre code, nous avons effectué pendant toute la durée du projet, différents tests.
A chaque création de nouvelle fonction, nous l'avons testé. Par exemple, nous nous sommes assuré que les fonctions HumidificationSol et AdaptationSol définies dans la classe Terrain fonctionnaient comme nous le voulions puis nous avons fait la même chose avec les fonctions GererLumiere et GererTemperature.
Après la définition de nos différentes plantes, nous avons également fait de nombreux tests pour tester que toutes les plantes pouvaient grandir et mourir. Nous avons ainsi observé ce qu'il se passait en changeant la météo, la saison, ainsi que le nombre de voisins.
Nous avons ensuite fait des tests pour vérifier le bon fonctionnement des fonctions dans Simulation comme l'affichage du plateau et son agrandissement.
A la suite de cela, nous avons effectué tous nos tests en utilisant la classe Simulation puisque le jeu pouvait à partir de là tourner seul.

V. La modélisation objet réalisée

Notre jeu comporte plusieurs classes réparties dans plusieurs fichiers C#. Les principales classes sont:

- La classe Plante
- La classe Terrain
- La classe Simulation

Il y a de plus la classe Ratiboiseur permettant de gérer l'animal nuisible mais elle est moins importante que les autres, car le jeu peut fonctionner sans elle.

Classe Plante

La classe Plante est une classe abstraite regroupant tous les types de plantes existantes dans notre jeu. De cette classe sont définies d'autres classes abstraites permettant de séparer au mieux les plantes de notre univers. Ces catégories sont :

- les plantes comestibles
- les plantes commercialisables
- les plantes ornementales
- les plantes invasives

Ces classes existent en vue d'implémenter des fonctionnalités de vente et de consommation dans l'optique de bonus pour le joueur. Ces fonctionnalités n'ont pas été implémentées dans notre code due à des contraintes de temps et de complexité du code.

Les différents types de plantes instanciables dans le programme remplissent les paramètres nécessaires à la création de plantes automatiquement lors de la création d'une instance.

Classe Terrain

La classe abstraite Terrain regroupe tous les types de terrain utilisables possibles de notre jeu. Contrairement aux plantes, les classes instanciables sont directement héritées de la classe Terrain. De la même manière que les plantes, les différents terrains remplissent les paramètres nécessaires dans la classe Terrain.

Les classes Plante et Terrain sont reliées entre elles par un paramètre dans chacune. Ainsi, on peut accéder facilement à la plante depuis une instance de terrain et inversement. Ce procédé est important afin de communiquer les valeurs du terrain à la plante pour calculer sa capacité à grandir.

Classe Simulation

La classe Simulation est la classe principale utilisée pour jouer au jeu. Elle regroupe les méthodes principales pour effectuer les boucles de jeu.
Cette classe contient notamment des listes de terrains et de plantes pour effectuer un affichage pour le joueur. A chaque ajout de terrain ou de plante par le joueur, ces listes sont mises à jour. De plus, le lien entre terrain et plante déclaré précédemment permet de procéder de manière quasiment automatique à la croissance des plantes.

Classe Ratiboiseur

La classe Ratiboiseur est instanciée une fois dans la simulation. Cette instance est activée lors d'un cas d'urgence, décidé de manière aléatoire par la simulation.

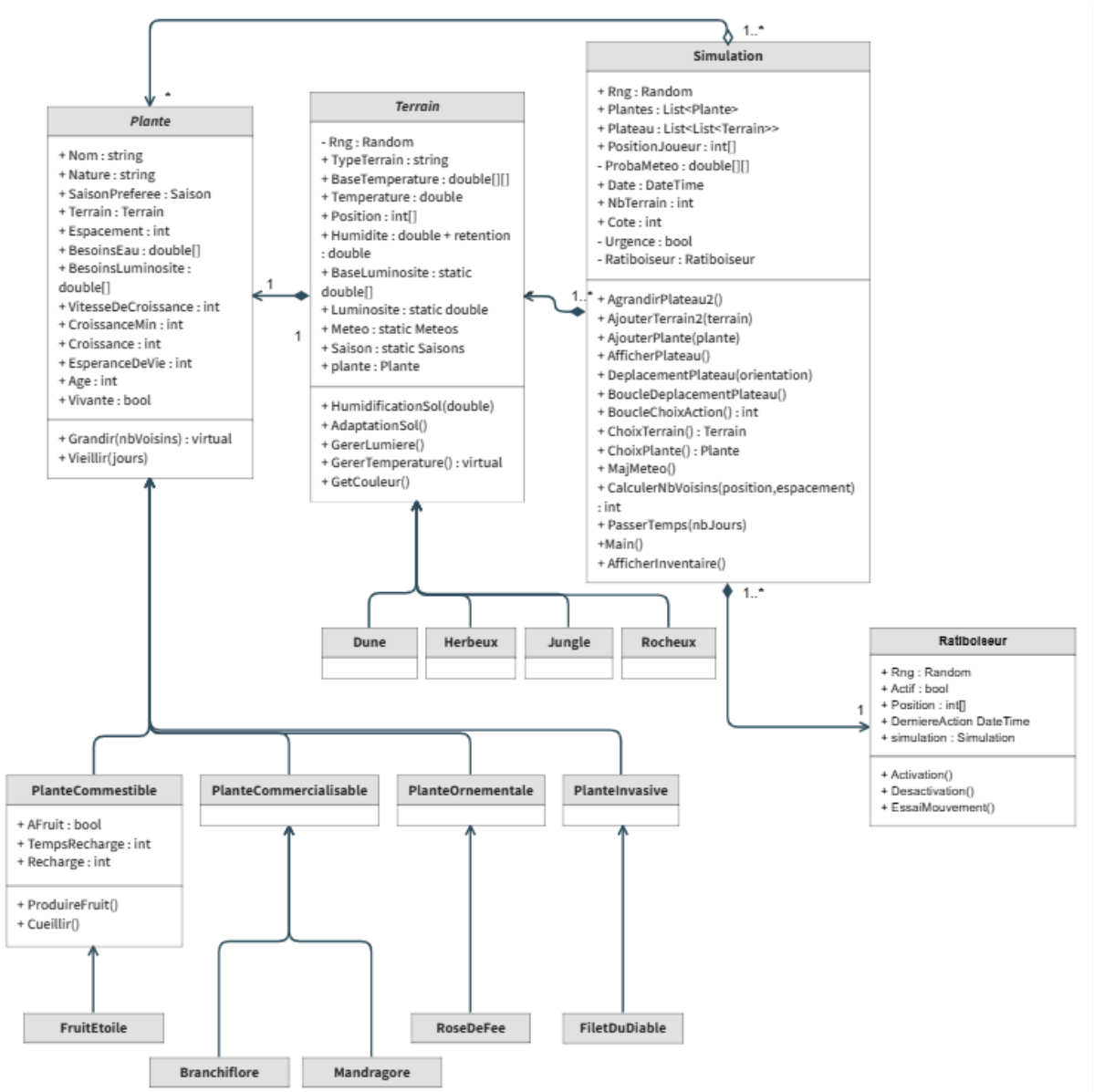


Diagramme UML représentant les classes de notre projet

VI. Gestion de projet

Lors de la première séance de travail, nous avons commencé par lister sur papier les différentes classes et fonctions que notre projet allait nécessiter. Nous avons également défini l'univers dans lequel nous voulions que notre jeu se déroule. Une fois, les différentes tâches, nous nous les sommes réparties. Hugo a commencé à créer la classe Terrain et Emilie la classe Plante. Nous avons ainsi avancé chacun de notre côté. Nous avons fait des points régulièrement pour suivre l'avancement global et se répartir les nouvelles choses à faire. Nous nous sommes également entre-aidés en cas de problèmes.

VII. Bilan critique sur le projet

A- Problèmes rencontrés

Au commencement du projet, nous avons eu des difficultés à définir l'univers dans lequel nous voulions que notre jeu prenne vie. Nous avons donc eu du mal à démarrer. Nous avons ensuite eu des petites difficultés comme la gestion d'éléments communs à toutes les plantes et à tous les terrains, mais nous sommes parvenus à les dépasser. Notre problème principal a été l'envergure du projet. En effet, il y avait beaucoup d'éléments à penser et à coder, des éléments qui pour certains étaient liés entre eux. Nous n'avons ainsi pas eu le temps de faire toutes les fonctionnalités que nous aurions souhaité. Nous aurions, en effet, aimé mettre en place, un système de magasin avec de l'achat et de la vente de plantes et de fruits.

B- Impressions finales

Ce projet nous a permis de mettre en pratique ce que nous avions vu en cours. Nous avons dû à la fois travailler avec des éléments qui restaient les mêmes entre les classes, d'autres qui devaient être modifiés, ce qui nous a permis de voir les différents aspects de la programmation orientée objet, ce que nous avons apprécié. Nous avons également aimé être libres de choisir notre univers, même si nous avons eu des difficultés à nous décider.

Annexes

- Tests effectués dans Program.cs

```
using System.Diagnostics;

Meteo meteo = Meteo.Normal;
Saison saison = Saison.Printemps;

Terrain.meteo = meteo;
Terrain.saison = saison;

Terrain terr1 = new Herbeux();
Terrain terr2 = new Rocheux();
Terrain terr3 = new Dune();
Terrain terr4 = new Jungle();

// Vérifier que la MAJ de la luminosité, la température et de l'humidité marchent bien

// double baseHumidite = 0.5;
// terr1.HumidificationSol(baseHumidite);
// terr1.AdaptationSol();
// Console.WriteLine(terr1.humidite);
// Debug.Assert(terr1.humidite == baseHumidite - (1 - terr1.retention) * 0.01);

// terr2.HumidificationSol(baseHumidite);
// terr2.AdaptationSol();
// Console.WriteLine(terr2.humidite);
// Debug.Assert(terr2.humidite == baseHumidite - (1 - terr2.retention) * 0.01);

// terr3.HumidificationSol(baseHumidite);
// terr3.AdaptationSol();
// Console.WriteLine(terr3.humidite);
// Debug.Assert(terr3.humidite == baseHumidite - (1 - terr3.retention) * 0.01);

// terr4.HumidificationSol(baseHumidite);
// terr4.AdaptationSol();
// Console.WriteLine(terr4.humidite);
// Debug.Assert(terr4.humidite == baseHumidite - (1 - terr4.retention) * 0.01);

terr1.GererLumiere();

// for (int i = 0; i < 20; i++)
// {
```

```
//     terr1.GererLumiere();
//     Console.WriteLine(Terrain.luminosite);
//     Debug.Assert(Terrain.luminosite < 0.75 + 0.1 && Terrain.luminosite > 0.75 - 0.1);
// }

// for (int i = 0; i < 20; i++)
// {
//     terr1.GererTemperature();
//     Console.WriteLine(terr1.temperature);
//     Debug.Assert(terr1.temperature < 18.0 + 1 && terr1.temperature > 18.0 - 1);
// }
// Console.WriteLine();
// for (int i = 0; i < 20; i++)
// {
//     terr2.GererTemperature();
//     Console.WriteLine(terr2.temperature);
//     Debug.Assert(terr2.temperature < 18.0 + 1 && terr2.temperature > 18.0 - 1);
// }
// Console.WriteLine();
// for (int i = 0; i < 20; i++)
// {
//     terr3.GererTemperature();
//     Console.WriteLine(terr3.temperature);
//     Debug.Assert(terr3.temperature < 22.0 + 1 && terr3.temperature > 22.0 - 1);
// }
// Console.WriteLine();
// for (int i = 0; i < 20; i++)
// {
//     terr4.GererTemperature();
//     Console.WriteLine(terr4.temperature);
//     Debug.Assert(terr4.temperature < 19.0 + 1 && terr4.temperature > 19.0 - 1);
// }


// Créer des plantes
Plante plante1 = new Mandragore(terr1); //ok
Plante plante2 = new Branchiflore(terr2); //ok
Plante plante3 = new FruitEtoile(terr3); //ok
Plante plante4 = new FiletDuDiable(terr4); //ok
Plante plante5 = new RoseDeFee(terr1); //ok


// Console.WriteLine(plante1);
// Console.WriteLine(plante2);
// Console.WriteLine(plante3);
```

```
// Console.WriteLine(plante4);
// Console.WriteLine(plante5);


// Vérifier que la croissance de la plante fonctionne
//Terrain.saison = Saison.Ete;
//terr1.HumidificationSol(-50.0);
//plante1.Grandir(1);
// Console.WriteLine(terr1.humidite);

// terr3.GererLumiere();
// Terrain.saison = Saison.Automne;
// terr3.HumidificationSol(0.05);
// Terrain.meteo = Meteo.Soleil;
// plante3.Grandir(1);
// Console.WriteLine(Terrain.luminosite);
// Console.WriteLine(terr3.humidite);

// Terrain.saison = Saison.Ete;
// Terrain.meteo = Meteo.Orage;
// terr4.GererLumiere();
// terr4.HumidificationSol(0.25);
// plante4.Grandir(1);
// Console.WriteLine(Terrain.luminosite);
// Console.WriteLine(terr1.humidite);



Terrain.saison = Saison.Ete;
Terrain.meteo = Meteo.Soleil;
terr1.GererLumiere();
terr1.HumidificationSol(0.1);
plante5.Grandir(0);


terr1.plante = plante1;
terr2.plante = plante2;
terr3.plante = plante3;
terr4.plante = plante4;

Simulation simulation = new Simulation();
simulation.AfficherPlateau();
System.Threading.Thread.Sleep(1000);
simulation.plateau[0][0] = terr1;
simulation.plateau[0][1] = terr2;
```

```
simulation.plateau[1][0] = terr3;
simulation.plateau[1][1] = terr4;
simulation.AfficherPlateau();
System.Threading.Thread.Sleep(1000);

simulation.AgrandirPlateau(false);
simulation.AgrandirPlateau(true);
simulation.AfficherPlateau();
System.Threading.Thread.Sleep(1000);
simulation.positionJoueur = [1, 0];
simulation.AfficherPlateau();
System.Threading.Thread.Sleep(1000);
```