

## Projet de programmation avancée

### Conception du jeu ENSemenC



### Document de justification technique

LEFRAIS Jeanne  
PEIGNÉ Louison

Promotion 2027 - 1A - Groupe 1

# Table des matières

## Table des matières

|  |           |
|--|-----------|
|  | <b>1</b>  |
| <b>1. Introduction</b>                                     | <b>2</b>  |
| 1.1. Contexte  | 2         |
| 1.2. Enjeux techniques                                     | 2         |
| 1.3. Présentation générale du jeu                          | 3         |
| <b>2. Gestion de projet</b>                                | <b>4</b>  |
| <b>3. Structuration du code</b>                            | <b>6</b>  |
| 1.1. Les plantes   | 6         |
| 1.1.1. Conception orientée objet des plantes               | 7         |
| 1.1.2. Initialisation des plantes sur le terrain           | 7         |
| 1.1.3. Gestion de la croissance et de la santé des plantes | 8         |
| 1.2. Les terrains  | 8         |
| 1.3. Les fonctionnalités (actions)                         | 9         |
| 1.3.1. Vue principale sur le potager                       | 9         |
| 1.3.2. Déplacements vers des bâtiments                     | 9         |
| • La maison  | 9         |
| • Le Cabanon   | 10        |
| • La Grange  | 10        |
| 1.3. La dynamique spatio temporelle                        | 10        |
| 1.4. L'affichage   | 10        |
| <b>4. Tests et vérifications</b>                           | <b>13</b> |
| <b>5. Futures évolutions du jeu</b>                        | <b>13</b> |
| 5.1. Problèmes à résoudre                                  | 13        |
| 5.2. Améliorations potentielles                            | 14        |
| <b>6. Bilan général du projet</b>                          | <b>14</b> |

# 1. Introduction

## 1.1. Contexte

Le projet ENSemenC s'inscrit dans le cadre du cours de programmation avancée dispensé à l'ENSC en 2025. Ce projet a pour but de concevoir et de programmer un simulateur de potager en mode console, intégrant les principes fondamentaux de la programmation orientée objet en C#. Il constitue une suite logique aux projets précédents, mais cette fois avec une montée en complexité par l'utilisation systématique de concepts comme l'héritage, le polymorphisme, les classes abstraites ou encore les énumérations, conformément aux attendus du cours.

Nous avons disposé de plusieurs séances encadrées pour amorcer ce projet, complétées par un important travail personnel. Travaillant en binôme, nous avons commencé par imaginer les contours de notre simulateur et structurer notre code à l'aide du patron de conception MVC, afin de bien distinguer les données, les actions et l'affichage. Comme pour notre précédent projet, nous avons élaboré un plan de code clair, enrichi de schémas et d'exemples de logique métier, avant de nous lancer dans l'implémentation.

Les échanges entre nous ont été constants tout au long du projet. Une partie du temps a été consacrée à la structuration des classes, à la définition des règles du jeu et à la répartition équitable des tâches, mais nous avons aussi régulièrement travaillé à deux sur certaines parties plus complexes, comme la gestion des conditions climatiques ou les interactions entre le joueur et les plantes. Enfin, l'usage de GitHub s'est imposé naturellement comme outil central pour suivre l'évolution du projet, gérer les différentes versions, et effectuer les tests croisés nécessaires à la fiabilité du code.

## 1.2. Enjeux techniques

Le projet ENSemenC a posé plusieurs défis techniques qui ont fortement influencé notre manière de coder. Tout d'abord, nous avons dû concevoir une architecture orientée objet complète à partir de zéro. Cela impliquait de définir avec précision des classes abstraites (**Plante**, **Terrain**) et leurs spécialisations, de gérer l'association logique entre objets (par exemple, l'affectation de plusieurs plantes à un même terrain), et d'assurer un bon équilibre entre généralité et spécificité dans les comportements (comme la croissance ou la récolte).

Un autre enjeu important a été la simulation du temps et de la météo : à chaque tour, des conditions climatiques aléatoires (ensoleillement, pluie, température) sont générées et viennent impacter la croissance des plantes selon leurs préférences. Nous avons dû modéliser ces interactions de manière réaliste mais compréhensible, en définissant des seuils de tolérance, des effets de stress sur les plantes, et des conséquences à court ou moyen terme. Ces règles de jeu ont été implémentées dans la logique métier, tout en restant lisibles et modifiables.

D'un point de vue technique, nous avons veillé à respecter les conventions C#, à éviter la duplication de code, et à commenter les sections importantes. De plus, nous avons maintenu une structure de fichiers claire, avec un fichier `Program.cs` minimal, un contrôleur principal (`JeuController`), et des classes modèles séparées. GitHub nous a permis de travailler en parallèle via des branches, de documenter nos commits, et de revenir en arrière en cas d'erreurs, ce qui a considérablement amélioré notre efficacité et notre organisation.

### 1.3. Présentation générale du jeu

ENSEmenC est un simulateur dans lequel le joueur incarne un jardinier responsable d'un potager interactif réparti sur plusieurs terrains. Chaque plante possède ses propres caractéristiques biologiques : saison de semis, type de sol préféré, besoin en eau, lumière, température, maladies possibles, espérance de vie, etc. Le joueur doit veiller à adapter chaque plantation à son terrain, puis à l'entretenir au fil des tours de jeu en réagissant aux changements de météo, aux urgences, et aux événements aléatoires.

Le jeu comporte deux modes : un mode classique, où chaque tour correspond à une semaine, et où le joueur peut semer, arroser, désherber, pailler ou récolter ; et un mode urgence, déclenché aléatoirement, qui simule l'irruption d'un intrus ou une intempérie violente (tempête, grêle, sécheresse...). Dans ce second mode, le joueur dispose de quelques secondes pour prendre une décision stratégique rapide (déployer une bâche, faire du bruit, fermer une serre...) afin de protéger ses récoltes.

Le jeu est entièrement en console, avec un affichage simplifié mais dynamique : les plantes sont représentées par des symboles, les terrains sont placés sur une grille, et un personnage-joueur se déplace librement entre les zones du jardin. Chaque action a un impact direct sur les plantes, dont l'état évolue à chaque tour. L'objectif est de maintenir un potager sain le plus longtemps possible, en

optimisant les ressources naturelles (pluie, soleil), en anticipant les menaces, et en réagissant aux aléas du vivant.

Ce projet nous a permis de donner vie à un univers végétal cohérent, en reliant les apports techniques du cours de C# à une mécanique de jeu progressive, intuitive, et modulable. Il s'agit d'un jeu à la fois pédagogique, stratégique et immersif, dans lequel les choix du joueur influencent directement l'évolution de son environnement.

## 2. Gestion de projet

Le choix du sujet s'est fait en lien avec un autre projet mené en parallèle, dans lequel nous devions construire une ontologie autour des cocktails. Cette coïncidence nous a semblé être une bonne occasion de créer une synergie entre les deux projets : nous avons donc décidé de concevoir un potager dont les plantes serviraient à composer des recettes de cocktails. Nous avons ainsi choisi 6 plantes permettant la confection de mojitos et de piña colada. Avant de commencer à coder, nous avons constitué un inventaire détaillé des plantes et de leurs caractéristiques sur un tableau collaboratif, qui a dû être modifié au moment de coder, notamment afin de faire correspondre les échelles et unités entre les plantes et les terrains.







| Boisson :              | Mojito  |   |   | Piña colada   |   |   |
|------------------------|---|---|---|---|---|---|
| Plantes nécessaires :  | Canne à sucre   | Menthe  | Citron vert   | Cocotier  | Cerisier  | Ananas  |
| Nature                 | Comestible / Vivace   | Comestible / Vivace   | Comestible / Vivace   | Comestible / vivace   | Comestible / vivace   | Comestible / vivace   |
| Saison de semis        | Septembre   | Avril   | Mai   | Mars  | Février   | Juin  |
| Terrain préférable     | Sableux à côté source d'eau   | sol bien drainé, humide   | drainé, fertile   | brochure de mer, sable pauvre, soleil et vent salé                                  | terrains calcaires  | sol sableux, bien drainé  |
| Espacement             | 120cm   | 35 cm entre chaque plant  | 4m entre chaque citronnier  | 80cm  | 5m  | 75 cm   |
| Place nécessaire       | 20cm  | 50 cm   | 17m   | 7,5m  | 10m   | 1m  |
| Vitesse de croissance  | lent (1an)  | rapide-quelques jours après plantation  | lent - 45 cm par an   | très lent (7 à 10 ans pour avoir des noix)  | moyenne/ rapide   | lente - 2 ans   |
| Besoin eau             | 1500 mm/ an   | 1000mm/an   | 1000mm/an   | humidité constante 60%  | 1300 m3 par an  | 900mm/an  |
| Besoin lum             | big soleil big chaleur  | mi-ombre, soleil doux   | exposition au soleil pendant plusieurs heures                                       | vive lumière indirecte  | beaucoup d'ensoleillement, 0 ombre si possible  | exposition au soleil  |
| Température pref       | minimum 16°C  | 20°C  | 25°C  | 30°C  | 20°C  | 26°C  |
| Maladies attrapables   | mildiou, rouille (champi) / gommose (bactéries)                                     | rouille (champi) / mildiou  | Gommose (bactéries) / mildiou   | bipolaris   | champi  | Fusariose   |
| Espérance de vie       | 5-10 ans  | 4 ans   | 40 ans  | 100 ans   | 50 ans  | 5 ans   |
| Nbr de fruits/ pousses | 180g sucre/ tige  | 15 pousses par récolte  | 150 fruits par récolte  | 35 noix par récolte   | 300 fruits par récolte  | un seul fruit   |
|                        |  |  |  |  |  |  |

Figure 1 : Tableau récapitulatif des caractéristiques de nos plantes

La suite du projet s'est construite autour d'une répartition des tâches équilibrée. Louison s'est chargée de la classe abstraite Plante et des classes héritières qui en découlent, de la classe Meteo et de l'interface (affichages). Jeanne a pris en main les classes Terrain, le développement du mode urgence, et nous avons codé ensemble les classes Actions et Effets, assuré la cohérence entre les classes, et commenté le code à deux. La rédaction du présent compte-rendu a elle aussi été partagée.

Si cette répartition nous a permis d'avancer efficacement en parallèle, certaines difficultés (notamment les interactions entre classes ou les effets dynamiques du climat) nous ont poussées à collaborer plus étroitement à certains moments. Cela a renforcé notre compréhension globale du projet et enrichi notre maîtrise de la programmation orientée objet. Malgré les contraintes de temps, ce projet s'est avéré particulièrement formateur et nous a permis de consolider nos compétences techniques tout en développant un jeu original autour d'un thème qui nous a motivées.

### 3. Structuration du code

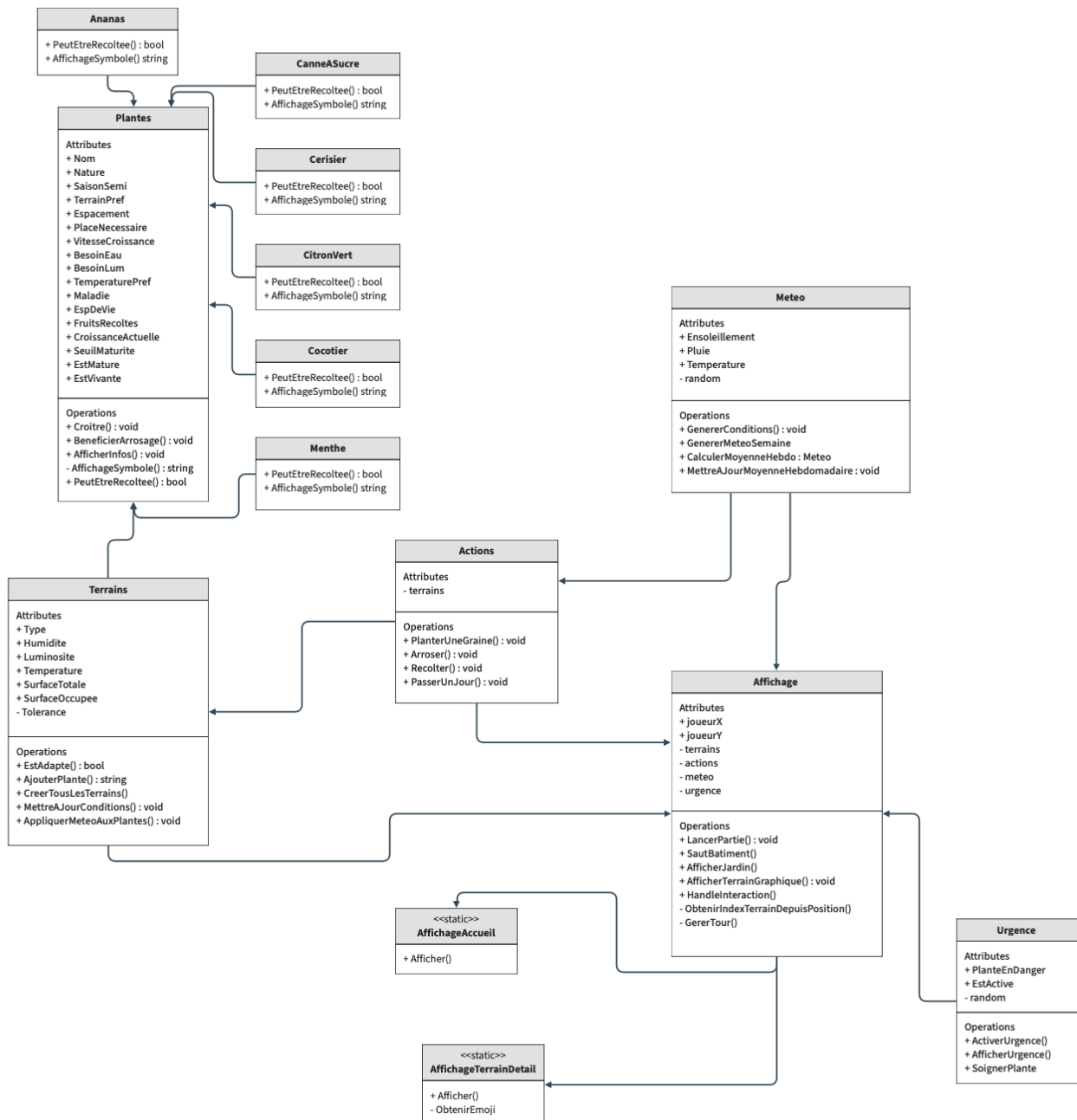


Figure 2 : Diagramme UML

#### 1.1. Les plantes

Dans notre jeu de gestion de potager, les plantes occupent une place centrale à la fois sur le plan fonctionnel et immersif. Afin de modéliser les spécificités de chaque plante et de faciliter les évolutions du jeu, nous avons conçu une architecture orientée objet reposant sur une classe mère

Plante, dont héritent plusieurs classes spécifiques telles que Menthe, Cocotier, Ananas, Cerisier, CitronVert et Canne à sucre. Ce choix permet d'encapsuler à la fois les caractéristiques communes à toutes les plantes (croissance, état de santé, besoin en eau ou en soleil) et les comportements particuliers liés à chaque espèce (résistance à certaines maladies, production, rendement, terrain préféré).

### 1.1.1. Conception orientée objet des plantes

La classe Plante joue le rôle de socle commun à toutes les espèces végétales. Elle contient les propriétés générales suivantes :

- Nom de la plante (pour l'affichage et la lisibilité),
- Stade de croissance (semence, jeune pousse, plante adulte, plante fanée)
- Besoin en eau et en soleil, représentés sous forme de compteurs,
- Résistance à certaines maladies,
- Terrain préféré, pour encourager des décisions stratégiques de placement.

À partir de cette structure, nous avons créé plusieurs classes dérivées pour représenter des espèces concrètes, chacune ayant des valeurs spécifiques pour les attributs susmentionnés. Par exemple, la classe Menthe préfère les zones ombragées et humides, pousse rapidement, mais est très sensible au mildiou, tandis que le Cocotier nécessite beaucoup de soleil et de temps pour atteindre sa maturité, mais produit une grande quantité de ressources une fois adulte.

Cette hiérarchisation par héritage nous a permis de centraliser le code de gestion des plantes dans la classe mère, tout en laissant aux classes filles la possibilité de redéfinir certains comportements via des méthodes virtuelles (comme la méthode Croître() ou RéagirAuxIntempéries()), ce qui favorise l'extensibilité du projet.

### 1.1.2. Initialisation des plantes sur le terrain

Au lancement du jeu, une grille de terrains vides est générée. Aucun végétal n'est présent initialement : c'est au joueur de choisir s'il souhaite planter une graine sur une case spécifique. Cette approche volontaire favorise une gestion stratégique du potager, car chaque plantation dépend des décisions du joueur en fonction des ressources disponibles, de la météo, et des propriétés du terrain.



Le processus de plantation suit un ordre précis : le joueur commence par choisir la plante qu'il souhaite cultiver. Ensuite, il sélectionne la parcelle de terrain sur laquelle il envisage de l'implanter. À ce moment-là, des informations lui sont fournies sur les caractéristiques des différents terrains disponibles, afin d'identifier celui qui est le plus adapté à la plante choisie, selon ses préférences en matière d'humidité, d'exposition ou de type de sol.

### 1.1.3. Gestion de la croissance et de la santé des plantes

Pour simuler le cycle de vie des plantes, nous avons mis en place un système de croissance par tour. À chaque tour, une méthode `Croître()` est appelée pour chaque plante, prenant en compte les conditions environnementales (ensoleillement du jour, niveau d'eau du terrain, météo générale). Si les besoins de la plante sont satisfaits, son stade de croissance progresse. En cas de manque ou d'excès, un système de détérioration de la santé peut s'activer, menant à un ralentissement de la croissance, voire à la mort de la plante.

Ce comportement est complété par l'introduction de maladies, contre lesquelles certaines plantes ont une résistance naturelle, définie dans leur classe spécifique. Ce système de santé dynamique encourage le joueur à observer attentivement ses cultures et à adapter ses actions (arrosage, rotation des cultures, lutte biologique, etc.).

## 1.2. Les terrains

Dans notre jeu, les terrains représentent les parcelles cultivables sur lesquelles les plantes peuvent être implantées. Nous avons modélisé six terrains en tout, et chacun est doté de propriétés spécifiques : type de sol, humidité, exposition au soleil, ou encore niveau de fertilité. Ces attributs influencent directement la croissance et la santé des plantes qui y sont installées, et renforcent ainsi la dimension stratégique du placement.

Initialement, nous avons envisagé une hiérarchie de classes avec une classe mère `Terrain` et des classes filles spécialisées (ex. : `TerrainSableux`, `TerrainHumide`, etc.). Toutefois, en pratique, ce modèle s'est révélé peu adapté à nos besoins fonctionnels. Pour simplifier l'implémentation et favoriser la souplesse du code, nous avons finalement opté pour une seule classe `Terrain`, non abstraite, avec des attributs modulables. Ce choix nous a permis de manipuler facilement des listes

de terrains et de rendre l'interfaçage avec les autres composants du jeu plus fluide, sans surcharge inutile liée à la gestion d'un héritage complexe.

### 1.3. Les fonctionnalités (actions)

Notre jeu propose un ensemble d'actions interactives réparties selon les lieux dans lesquels le joueur peut se déplacer : la maison, le cabanon, la grange et le potager principal. Chacun de ces lieux propose des fonctionnalités spécifiques, renforçant la dimension stratégique du jeu et encourageant l'exploration des différentes facettes de la gestion de potager.

#### 1.3.1. Vue principale sur le potager

Depuis l'écran principal, le joueur visualise les différents terrains disponibles, chacun pouvant accueillir une ou plusieurs plantes. Il peut planter une graine sur un terrain vide en choisissant l'espèce désirée, consulter le détail d'un terrain via une interface dédiée qui affiche les informations clés sur chaque plante (nom, type, stade de croissance, état de santé, maladies éventuelles et possibilité de récolte), et récolter les plantes arrivées à maturité pour obtenir des ressources. Le joueur surveille également l'état général des cultures pour décider d'arroser, soigner ou faire tourner les plantations. En cas d'événement d'urgence, comme une maladie soudaine, une alerte est déclenchée et demande une intervention rapide. En cliquant sur la touche "espace", le joueur accède au centre opérationnel du potager, où toutes les actions essentielles sont regroupées : planter, arroser, soigner et récolter. Ce lieu centralise la gestion active des plantes pour assurer leur santé et la productivité du potager.

#### 1.3.2. Déplacements vers des bâtiments

Le joueur peut à tout moment quitter la vue du potager pour se rendre dans un des bâtiments disponibles, chacun correspondant à une fonction clé dans la gestion de son jardin :

- **La maison**

La Maison (Home) est un lieu de repos permettant au joueur de faire avancer le temps : en s'y rendant, il simule automatiquement une semaine entière, ce qui fait progresser la croissance de toutes les plantes en fonction d'une météo quotidienne générée aléatoirement (ensoleillement, pluie, température). Une moyenne hebdomadaire est ensuite calculée et affichée, offrant au joueur une vision claire des conditions climatiques ayant influencé son potager.

- **Le Cabanon**

Le Cabanon est l'atelier du jardinier, un lieu consacré à l'action concrète : en s'y rendant, le joueur peut planter une nouvelle graine sur un terrain de son choix, choisissant parmi les plantes disponibles. Il peut également effectuer diverses actions afin de prendre soin de ses plantes, comme les arroser, les traiter ou les récolter. S'il se trouve loin du cabanon, l'utilisateur peut appuyer sur la touche Espace comme raccourci clavier afin d'effectuer ces mêmes actions.

- **La Grange**

La Grange est le lieu de gestion des récoltes et du suivi global des plantes. En y entrant, le joueur peut consulter l'inventaire des plantes disponibles (menthe, citron vert, ananas, cerisier, cocotier, canne à sucre) ainsi que la liste des plantes actuellement plantées sur les terrains, indiquant simplement leur type et leur vitesse de croissance (ex : "Menthe (Croissance : Rapide)"). Si une plante est malade, une alerte apparaît et le joueur peut choisir de la soigner immédiatement pour éviter sa perte.

### **1.3. La dynamique spatio temporelle**

Le jeu fonctionne en mode tour par tour, où chaque tour correspond à une unité de temps variable selon le contexte : lorsqu'on passe une nuit dans la maison, cela équivaut à une semaine entière dans le temps du jeu. À chaque tour, la météo est générée aléatoirement et affecte tous les terrains par des paramètres tels que l'ensoleillement et les précipitations. Les plantes évoluent dans leur cycle de croissance en tenant compte des conditions locales, notamment leur espacement défini, qui influence leur développement en fonction de la place qu'elles occupent sur la grille. Des maladies peuvent apparaître de manière aléatoire et nécessitent une intervention rapide du joueur.

### **1.4. L'affichage**

L'affichage dans notre jeu de gestion de potager est géré principalement par la classe Affichage.cs, qui orchestre la représentation visuelle du jardin, des bâtiments, des terrains, ainsi que l'affichage des interactions et des éléments contextuels tels que la météo et les urgences.

Par ailleurs, la classe AffichageTerrainDetail.cs est dédiée à l'affichage détaillé d'un terrain spécifique lorsque le joueur interagit avec celui-ci. Elle présente notamment la liste des plantes présentes sur ce terrain, avec leurs caractéristiques précises telles que le nom, le stade de croissance,

l'état de santé (vivante ou malade), les maladies éventuelles, ainsi que la possibilité ou non de les récolter.

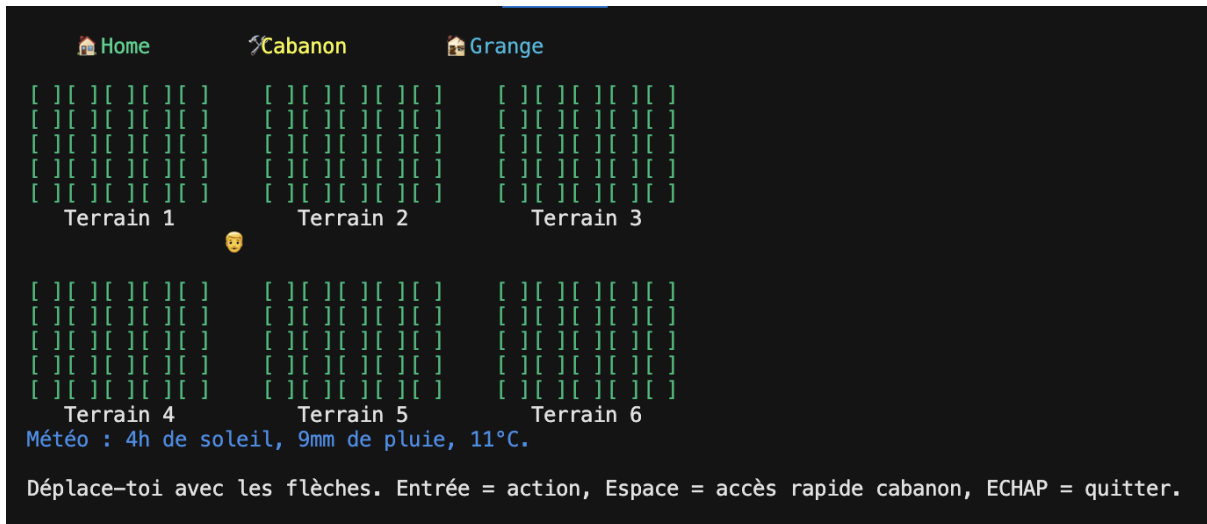
La classe AffichageAccueil.cs gère l'écran d'accueil du jeu. Elle affiche une introduction visuelle avec un logo en ASCII art ainsi que des instructions destinées au joueur, favorisant une immersion progressive avant le lancement de la partie. Elle joue un rôle clé dans l'introduction et la contextualisation de l'univers du jeu.



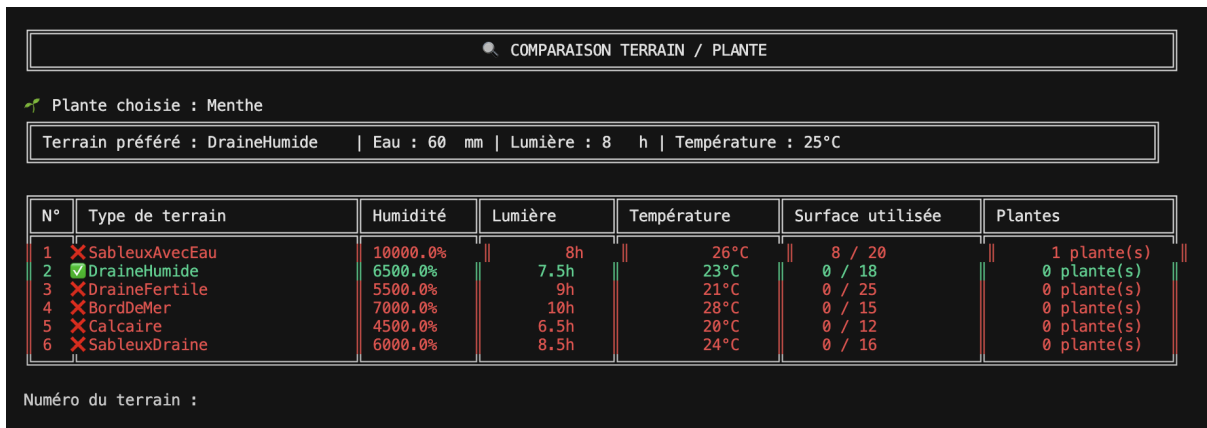
Figure 3 : Affichage de l'accueil du jeu



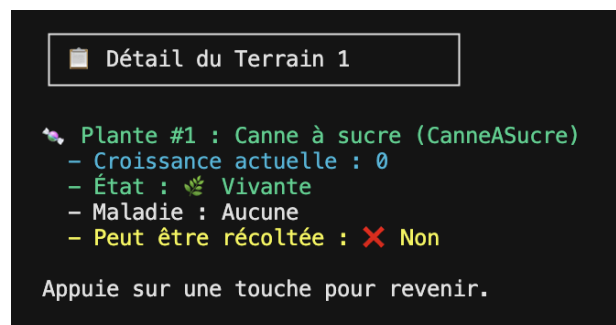
Figure 4 : Affichage des règles du jeu



*Figure 5 : Affichage du jeu, avec les bâtiments et les terrains*



*Figure 6 : Affichage de l'interface lorsque l'utilisateur souhaite planter*



*Figure 7 : Affichage du terrain 1 après avoir planté une Canne à sucre*

## 4. Tests et vérifications

Tout au long du développement, nous avons adopté une approche de tests itératifs manuels, consistant principalement à vérifier le bon fonctionnement de notre application par l’affichage à l’écran. Dès les premières étapes du projet, nous avons mis en place des messages de débogage et des impressions console pour suivre l’évolution des variables, valider les actions effectuées (comme la plantation ou l’arrosage), et nous assurer que les interactions entre les différentes classes étaient bien gérées. Cela nous a permis d’identifier rapidement les incohérences dans la gestion des états des plantes, les problèmes d’affichage sur le plateau, ou encore les erreurs liées aux conditions météo. Plutôt que de concevoir une suite de tests automatisés, ce qui aurait été trop long au vu du temps imparti et du reste de notre charge de travail, nous avons privilégié une méthode empirique, en testant régulièrement chaque nouvelle fonctionnalité ajoutée. Cette approche a été particulièrement utile pour détecter des erreurs de logique ou d’enchaînement d’actions, et a facilité les ajustements au fil de l’évolution du code. Même si cette méthode de test reste perfectible, elle s’est révélée suffisante pour assurer un fonctionnement globalement stable et cohérent du jeu.

## 5. Futures évolutions du jeu

### 5.1. Problèmes à résoudre

Malgré les nombreuses fonctionnalités déjà mises en place dans le jeu de gestion du potager, plusieurs aspects restent incomplets ou présentent des limitations qui impactent l’expérience de jeu. Un premier point concerne l’affichage du récapitulatif hebdomadaire dans la grange. Actuellement, les informations sont présentées jour par jour, ce qui peut alourdir la lecture et diluer les données essentielles. Il serait pertinent soit de regrouper ces informations en un résumé synthétique de la semaine, soit d’optimiser leur présentation en séparant clairement les journées de manière lisible.

Par ailleurs, la gestion de la mort des plantes est encore insuffisamment développée. Lorsqu’une plante dépérit, aucune conséquence concrète ou retour visuel n’est proposé, ce qui nuit à la cohérence du système de croissance et d’entretien. Un mécanisme clair de retrait, d’impact ou de notification serait nécessaire.

En outre, la grange elle-même pourrait être enrichie, tant sur le plan fonctionnel qu’informatif. Actuellement, son rôle se limite à un simple affichage, alors qu’elle pourrait devenir un véritable centre de suivi des cultures, avec des historiques, des conseils ou des prévisions.

Finalement, le mode urgence, bien qu'introduit, mérite d'être approfondi. La gestion des événements imprévus reste sommaire, et l'équilibrage de la difficulté pourrait être ajusté afin d'offrir une expérience stimulante mais maîtrisée.

## 5.2. Améliorations potentielles

Si nous avions disposé de plus de temps, nous aurions souhaité intégrer plusieurs fonctionnalités clés pour enrichir l'expérience utilisateur et rendre le jeu plus complet et immersif. Tout d'abord, une intégration complète de l'encyclopédie botanique dans le cabanon, proposant des informations détaillées sur chaque plante, telles que leurs exigences de culture, vulnérabilités, compatibilités et conseils de soin, aurait permis d'ajouter une dimension pédagogique essentielle au jeu.

Ensuite, la mise en place d'un système de magasin ou marché, permettant au joueur d'acheter des graines rares, des outils spécialisés ou des améliorations pour son potager, aurait apporté une dimension économique et stratégique, renforçant la profondeur du gameplay.

Par ailleurs, un enrichissement de la classe Urgence avec une variété d'effets plus réalistes, incluant à la fois des nuisibles (pucerons, champignons, insectes nuisibles) et des effets bénéfiques (présence de coccinelles, "fées protectrices"), aurait permis de mieux représenter les influences positives et négatives sur les plantes, rendant l'évolution du potager plus dynamique et crédible.

De plus, le développement d'un système d'aide et de tutoriels accessibles depuis la maison aurait facilité la prise en main du jeu et l'assimilation progressive de ses mécaniques complexes, améliorant ainsi l'accessibilité pour tous les profils de joueurs.

Enfin, la création d'un espace "bar" aurait offert au joueur la possibilité d'expérimenter des recettes à base de plantes récoltées, introduisant un aspect ludique et éducatif original, stimulant la créativité et l'exploration.

## 6. Bilan général du projet

Le projet ENSemenC, ambitieux et riche en fonctionnalités, s'est révélé à la fois formateur et exigeant. S'il nous a permis de consolider notre maîtrise de la POO en C# à travers la conception d'un

jeu structuré, il a aussi mis en lumière plusieurs difficultés. Le projet s'est parfois heurté à un manque d'anticipation sur la charge réelle de travail, avec des tâches plus complexes qu'imaginé. Le manque de temps – lié au fait qu'il ne s'agissait que d'un projet parmi d'autres – a limité nos possibilités de finalisation, d'optimisation et de tests poussés.

Certaines fonctionnalités restent à améliorer, comme l'affichage récapitulatif de la semaine ou l'inventaire de la grange (qui ne mentionne pas si une plante est morte). L'expérience utilisateur gagnerait aussi à être retravaillée, tant sur le plan de l'ergonomie que de l'intuitivité. Enfin, nous aurions aimé aller plus loin dans la dimension créative, par exemple en intégrant un bar permettant de transformer nos plantes en cocktails à vendre, prolongeant ainsi la logique ludique et économique du jeu.

Malgré ces limites, ce projet nous a appris à structurer un code bien plus complexe que le projet du semestre précédent, à collaborer efficacement, et à concevoir une architecture évolutive. Il constitue un socle solide pour de futures améliorations... à condition d'avoir un peu plus de temps (et d'énergie) à y accorder.