

Document de Justification Technique

ENSemenc



I. Introduction.....	2
II. Univers et spécificités du jeu.....	2
III. Déroulement d'une partie.....	2
IV. Modélisation objet.....	2
V. Tests réalisés.....	2
VI. Gestion de projet.....	2
VII. Bilan.....	2

I. Introduction

Dans le cadre du module Programmation Avancée du Semestre 6, le projet de programmation de était la réalisation d'un jeu sous forme de simulateur de potager. Ce projet s'est réalisé en binôme, le nôtre étant constitué de BOURELY Louis et de VIVEN Pierre.

Le jeu est réalisé en C# sous VS Code, en mode console et met en application la Programmation Orientée Objet.

L'objet de ce document est l'explication des règles du jeu, la justification du code et l'exposition de l'organisation de l'équipe.

II. Univers et spécificités du jeu

A. Univers du jeu

Bienvenue dans notre simulateur de potager, dans ce jeu, vous nommez votre monde et faites la rencontre de l'âme millénaire du potager: Optegar. Sous la forme d'un champignon fantastique, il vous présentera le monde dont vous êtes désormais le propriétaire.

Prenez le temps de découvrir les 5 terrains  dont vous serez le gestionnaire ainsi que les 15 plantes  que vous pouvez faire pousser. Cependant, méfiez-vous des particularités de certains terrains et de certaines plantes, vous pourriez être assez surpris.

Chaque semaine, vous aurez l'occasion de vous rendre au magasin , où vous pourrez acheter vos semis , vendre des plantes qui ont fini de pousser et que vous avez récoltés  ou encore acheter des items qui vous seront utiles au fil de la partie. Peut-être est-il intéressant d'en avoir certains en stock, pour éviter de mauvaises surprises...

Si vous cherchez au bon endroit, vous pourrez également trouver des wikis ou documentations qui vous seront d'une grande aide.

Notez également pour le bon déroulement de la partie que chaque tour correspond à une semaine et que la météo peut varier toutes les semaines un petit peu. Mais un bulletin météo  devrait vous permettre de prévenir si les conditions météorologiques devenaient mauvaises. Dans un souci de réalisme, toutes les 13 semaines vous changerez de saison et la météo des terrains changera grandement. Mais attention, rien n'assure que tous les terrains aient le même cycle de saison.

 Bon courage, et puisse la magie de la nature t'inspirer à chaque graine plantée !

B. Spécificités du jeu

Ce jeu de gestion de potager se distingue par plusieurs mécaniques de gameplay originales, orientées vers la stratégie, l'adaptation climatique et la connaissance fine des plantes et terrains. Voici les spécificités principales qui définissent l'expérience du joueur :



Meteo Dynamique par terrain

Chaque terrain possède une météo propre et indépendante : température, humidité, ensoleillement et précipitations évoluent chaque semaine. Un bulletin météo hebdomadaire permet au joueur d'anticiper les conditions à venir. Et toutes les 13 semaines, un changement de saison se produit, mais chaque terrain peut avoir un cycle saisonnier décalé, apportant une dimension stratégique supplémentaire.

Gestion multi-terrains avec contraintes climatiques

Le joueur doit gérer 5 terrains distincts, chacun ayant ses particularités climatiques et certains ayant des fonctionnements particuliers tels que de la jachère ou des explosions aléatoires. Certaines plantes poussent mieux sur certains terrains, tandis que d'autres peuvent être désavantagées par un climat inadapté. L'objectif est de placer les bonnes plantes sur les bons terrains tout en anticipant l'évolution du climat.

Economie de jeu

Le jeu comprend de la gestion d'une monnaie du jeu: la VerdaMoula, en effet le joueur possède une somme initiale de départ et c'est à lui de faire les bons choix pour faire fructifier sa monnaie et ne pas faire faillite.

Pour cela, chaque semaine, le joueur peut :

- acheter des semis
- vendre ses récoltes
- acheter des objets utiles ou défensifs (bêche, raticide non mortel, sécateur...)
- agrandir ses terrains (coûte en VerdaMoula)

Le joueur étant limité par ses ressources, il doit choisir soigneusement les objets qu'il achète ou les semis qu'il achète en fonction des prévisions météo et des dangers potentiels. Sur certaines saisons, certaines plantes seront bien plus intéressantes que d'autres. Par ailleurs, le jeu est codé pour être challengeant, ce qui explique la grande variation des coûts des semis mais aussi des items.

Interaction fine et diversité entre les plantes et les terrains

Chaque plante est sensible à quatre paramètres météo qui varient selon le terrain et la saison :

- la température
- l'humidité
- l'ensoleillement
- la pluie

Une plante peut très bien pousser une semaine et disparaître la suivante si le climat change. Le joueur doit donc surveiller et comprendre les affinités de chaque plante avec son environnement pour adapter ses actions en conséquence.

Mais pour réussir dans ce jeu, les choses sont plus complexes, il faut associer

intelligemment les 15 plantes parmi les 5 terrains en fonction des terrains favoris des plantes mais aussi de la météo.

En effet, nous avons une grande diversité de plantes qui ont des caractéristiques météorologiques différentes mais aussi des plantes qui ont en plus d'autres caractéristiques comme la possibilité de se répandre (plantes filantes) ou la nécessité d'être entretenu et taillé.

Cette diversité impose déjà d'avoir des stratégies pour jouer mais elle est démultipliée lorsque nous prenons en compte l'existence de 5 terrains dont 2 qui se comportent différemment de tous les autres. L'un nécessite de la patience, quand l'autre explose au fur et à mesure aléatoirement.

Documentation intégré pour guider le jeu

Le jeu intègre plusieurs wiki et documentations accessible, permettant au joueur de :

- Consulter les données de chaque plante
 - Consulter les données de chaque terrain
 - Consulter les données de chaque item
- Cela rend le jeu accessible et dynamique sans sacrifier la profondeur.

Possibilité de sauvegarder

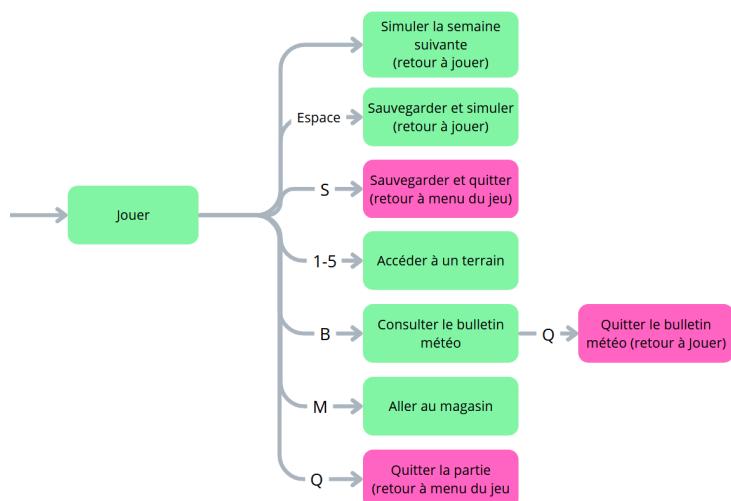
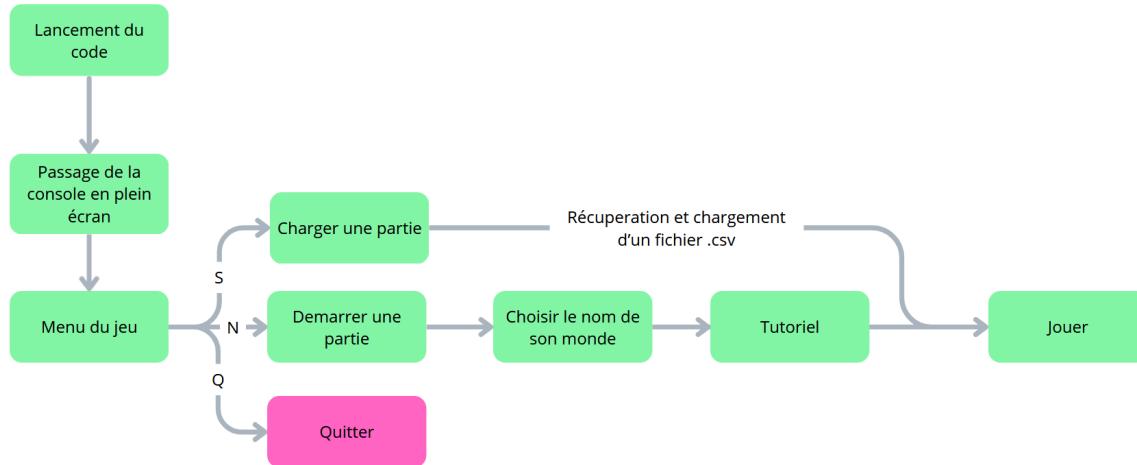
Le jeu rend possible la sauvegarde de la partie, en effet le jeu n'a pas de fin, c'est une simulation qui va de semaine en semaine et libre au joueur de s'arrêter lorsqu'il est satisfait (il me semble qu'il faut 100 000 VerdaMoula pour être satisfait). Dans cette dynamique, il nous semblait important que le joueur puisse sauvegarder sa partie et la reprendre lorsqu'il le souhaite.

Mode urgence

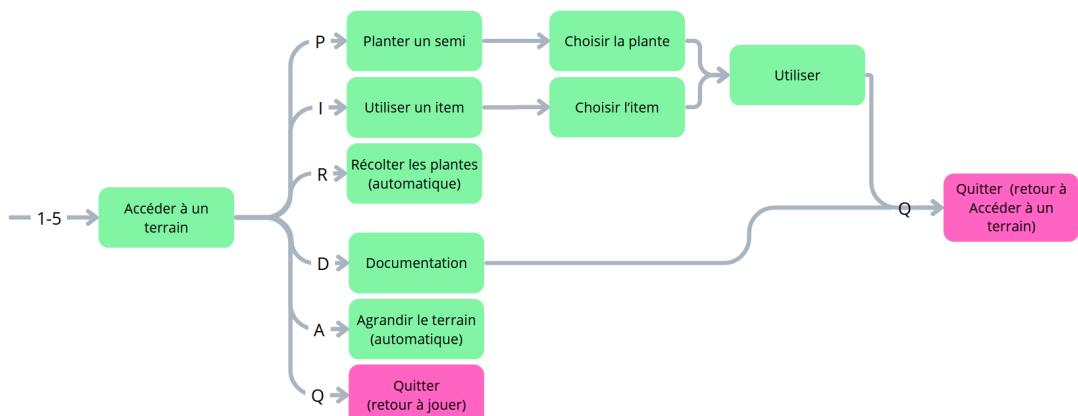
Le jeu contient aussi un mode urgence, c'est à dire que aléatoirement pendant la semaine, un événement urgent peut apparaître et être un bonus ou un malus. Dans le cas d'un malus, si le joueur possède l'item adapté, il pourra s'en prévenir, sinon il devra en subir les effets.

C. Possibilité du jeu

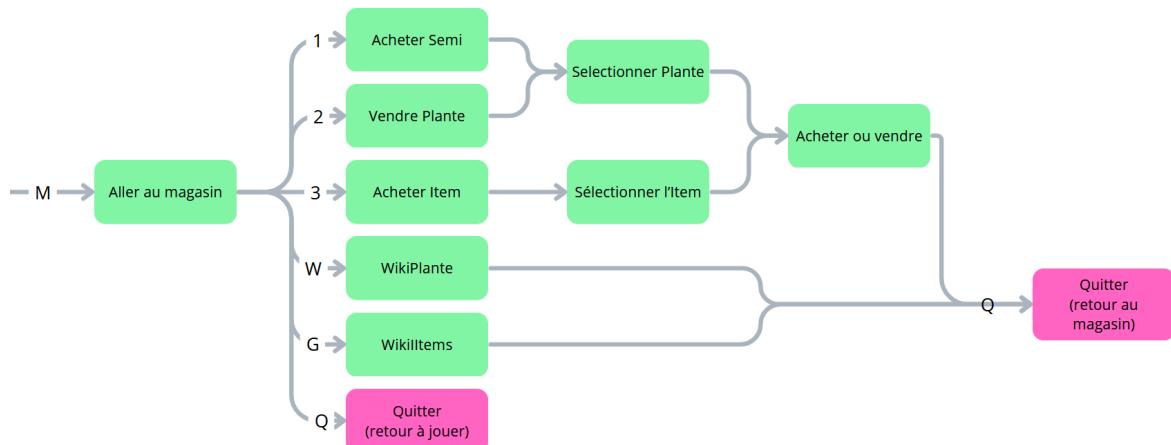
Dans cette partie, nous allons montrer sous la forme de graphique les différentes possibilités du jeu. La navigation est précisée sur le jeu.



Graphique du menu Jouer



Graphique du menu Terrain



Graphique du menu Magasin

III. Déroulement d'une partie

Étant donné la taille du code, expliquer chaque fonctionnalité ou chaque fonctionnement ne serait pas souhaitable dans le cadre d'un rapport technique comme celui-ci. Nous expliquerons alors simplement l'entrée, le fonctionnement des tours où nous jouons ainsi que les simulation des semaines.

Initiation du jeu :

En compilant le code avec dotnet run, il est demandé de mettre la console en plein écran, puis le menu débute et on nous propose soit de créer une nouvelle partie, soit de charger une partie déjà existante. Dans le premier cas, le joueur débute avec 100 Verdamoulas, 5 terrains partiellement débloqués, 0 plante, 0 item, 0 semi et entre dans une boucle de Jouer qui active Accueil(Partie) qui permet de faire un tour aussi longtemps que le joueur ne fait pas entrer. Lorsque le joueur clique sur entrer, la méthode SimulationSemaine() démarre activant la simulation de la semaine, là où nous pouvons voir apparaître les urgences. Dans le second cas, tout fonctionne pareil à la différence près que les valeurs initiales sont remplacées par les valeurs qui étaient enregistrées dans la sauvegarde.

Ainsi, on a donc un nombre infinis de semaines dans Accueil(Partie) et ses options de jeu, et ces semaines sont entrecoupées de simulation de semaine dans lesquelles apparaissent les urgences.

Program.cs

```

bool jeu = true;
while (jeu)
{
    Console.Clear();
    Afficher.Program();
  
```

```
switch (Console.ReadKey(true).Key) //Permet de choisir entre créer
un nouveau monde, récupérer une sauvegarde ou Quitter
{
    case ConsoleKey.N:
        Console.Clear();
        Console.Write("Tapez le nom de votre nouveau monde : ");
        string nom = Console.ReadLine()!;
        GestionJeu partieEnCours = new GestionJeu(nom);
        partieEnCours.Jouer(); //<= entrée boucle de Jouer
        break;
}
```

GestionJeu.cs

```
public void Jouer() //récupère une donnée de accueil et fait une action
en réponse
{
    bool enCours = true;
    while (enCours)
    {
        switch (Accueil(Partie)) //<= entrée boucle Accueil
        {
            case 1: //Semaine suivante
                SimulationSemaine(); //<= Demarre la simulation
                semaine
                break;
            case 2: //Sauvegarde et semaine suivante
                Sauvegarde sauvegarde = new Sauvegarde(Partie);
                sauvegarde.Sauvegarder();
                SimulationSemaine();
                break;
            case 3: //Sauvegarde et quitte
                Sauvegarde sauvegarde2 = new Sauvegarde(Partie);
                sauvegarde2.Sauvegarder();
                enCours = false;
                break;
            case 4: //Quitte sans sauvegarder
                enCours = false;
                break;
            case 5:
                Console.WriteLine("ERREUR @@@");
                break;
        }
    }
}
```



```
        default:  
            break;  
        }  
    }  
}
```

GestionJeu.cs

```
public int Accueil(Partie partie) //Accueil du jeu qui permet d'accéder  
aux différentes parties du jeu et de passer à la semaine suivante de  
sauvegarder ou quitter  
{  
    bool semaineEnCours = true;  
  
    while (semaineEnCours)  
    {  
        Console.Clear();  
        Afficher.Accueil();  
        ...  
        switch (Console.ReadKey(true).Key)  
        {  
            case ConsoleKey.Enter:  
                return 1; //<= Accueil(Partie)=1  
            ...  
        }  
    }  
}
```

GestionJeu.cs

```
public void SimulationSemaine()  
{  
    Partie.chenille = false;  
  
    Console.Clear();  
    Console.ForegroundColor = ConsoleColor.Gray;  
    Afficher.TexteEnProgressif("Simulation de la semaine ... ",  
    80);  
    Console.ForegroundColor = ConsoleColor.White;  
    Thread.Sleep(800);  
  
    Partie.Semaine++; //rajoute une semaine  
    Random random = new Random();  
}
```



```
        int urgences = random.Next(1, 31); //Génère une probabilité
d'une urgence Bonus/Malus
        switch (urgences)
        {
            case 10:
                Rat();
                break;
            case 11:
                Galinace();
                break;
            case 12:
                Chenille();
                break;
            case 13:
                FeeDesPlantes();
                break;
            case 14:
            case 15:
                AverseOR();
                break;
            default:
                break;
        }
        for (int k = 0; k < Partie.ListeTerrains.Length; k++) //Lance
VerificationTerrain dans chaque terrain pour faire simuler la
croissance des plantes
        {
            int saison = Partie.Semaine / 13 % 4;

Partie.ListeTerrains[k].VerifierTerrain(Partie.ListeTerrains[k],
saison);
        }
    }
```

Déroulement d'un tour/Accueil(Partie):

A chaque tour de jeu, le joueur est sur accueil et il a plusieurs options selon un switch case sur Console.ReadKey(true).Key. Pour les cases qui ont un return numérique, ça renvoie à Jouer(Partie) (présenté ci-dessus). Pour les autres, le nom de la méthode qui est appelée est parlante et ce sont des méthodes qui fonctionnent comme Accueil avec plusieurs actions gérées de la même manière.



GestionJeu.cs

```
    public int Accueil(Partie partie) //Accueil du jeu qui permet
d'accéder aux différentes parties du jeu et de passer à la semaine
suivante de sauvegarder ou quitter
    {
        bool semaineEnCours = true;

        while (semaineEnCours)
        {
switch (Console.ReadKey(true).Key)
{
    case ConsoleKey.Enter: //<=SimulerSemaine
        return 1;
    case ConsoleKey.S:      //<=Sauvegarder et quitter
        return 3;
    case ConsoleKey.Spacebar: //<=Sauvegarder et
simuler
        return 2;
    case ConsoleKey.Q:      //<=Quitter
        return 4;
    case ConsoleKey.B:      //<=Aller au bulletin Meteo
        BulletinMeteo();
        break;
    case ConsoleKey.D1:     //<=Aller au terrain 1
        VisualiserTerrain(Partie.ListeTerrains[0]);
        break;
    case ConsoleKey.D2:
        VisualiserTerrain(Partie.ListeTerrains[1]);
        break;
    case ConsoleKey.D3:
        VisualiserTerrain(Partie.ListeTerrains[2]);
        break;
    case ConsoleKey.D4:
        VisualiserTerrain(Partie.ListeTerrains[3]);
        break;
    case ConsoleKey.D5:
        VisualiserTerrain(Partie.ListeTerrains[4]);
        break;
    case ConsoleKey.M: //<=Aller au magasin
        Magasin();
}
```

```
        break;
    default:
        break;
    }
}
return 5;
}
```

Déroulement d'une semaine/SimulationSemaine()

Une fois toutes les actions réalisées depuis Accueil le joueur a la possibilité de SimulerSemaine() pour passer à la suivante.

```
Partie.Semaine++; //rajoute une semaine
```

Ensuite on génère un nombre aléatoire pour créer une possibilité de rentrer dans un des modes Urgences avec un Bonus ou Malus. Ces fonctions ont différents effets positifs ou négatifs et des items peuvent-être utilisés pour gérer ces situations.

```
Random random = new Random();
int urgences = random.Next(1, 31); //Génère une probabilité
d'une urgence Bonus/Malus
switch (urgences)
{
    case 10:
        Rat();
        break;
    case 11:
        Galinace();
        break;
    case 12:
        Chenille();
        break;
    case 13:
        FeeDesPlantes();
        break;
    case 14:
    case 15:
        AverseOR();
        break;
    default:
```



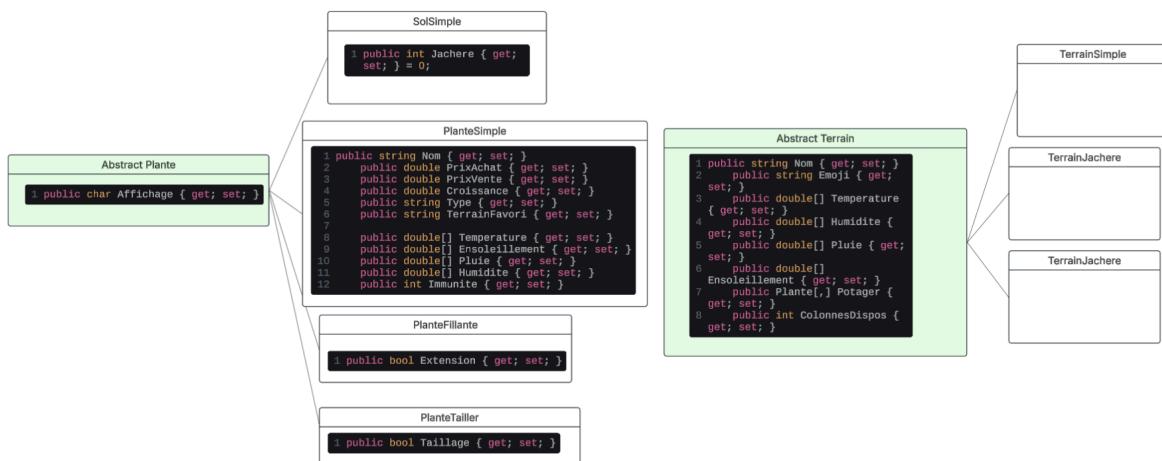
```
break;
```

Enfin dans chaque Terrain on utilise la méthode VerifierTerrain qui permet de SimulerCroissance() de chaque plante afin de les faire grandir ou mourir selon les conditions. Cette étape est essentielle, c'est elle qui fait tourner le jeu grâce aux méthodes des classes Terrain et Plante. Enfin une fois toutes les plantes simulées on peut passer à la semaine suivante et laisser au joueur découvrir les effets des changements de la simulation.

```
for (int k = 0; k < Partie.ListeTerrains.Length; k++) //Lance
VerificationTerrain dans chaque terrain pour faire simuler la
croissance des plantes
{
    int saison = Partie.Semaine / 13 % 4;

Partie.ListeTerrains[k].VerifierTerrain(Partie.ListeTerrains[k],
saison);
}
```

IV. Modélisation objet



Notre jeu possède au total 15 classes. Certaines sont très simples et d'autres plus complexes, héritent d'autres et sont primordiales. Ci-dessus les diagrammes des classes complexes pour mieux visualiser.

Terrain:

La classe Terrain abstract regroupe toutes les méthodes et attributs utiles pour les terrains. Notamment Potager qui est une matrice de Plante, une classe abstraite qui hérité de beaucoup de classe qui peuvent toutes se retrouver dans Potager. Trois classes héritent de Terrain :

- Terrain simple qui change juste le constructeur afin d'obtenir des réglages en fonction du terrain
- TerrainJachère qui réécrit VerificationTerrain afin de créer des SolSimple Jachère qui mettent un nombre aléatoire de semaine à repasser en SolSimple Laboure
- TerrainMine qui réécrit VerificationTerrain et génère une probabilité de faire exploser chaque case à chaque simulation

Plante:

La classe Plante abstract ne possède qu'un attribut et aucune méthode. c'est l'attribut Affichage qui est essentiel pour être affiché dans le Potager. Ensuite 4 classes héritent :

- SolSimple qui ne modifie que le constructeur pour choisir l'affichage en fonction d'un nom et qui ajoute l'attribut Jachère utilisé pour compter le nombre de semaines de Jachère. Sol Simple représente les plantes mortes non jouables.
- PlanteSimple qui ajoute tous les paramètres des plantes vivantes pour pouvoir les acheter les faire pousser et les vendre.
- PlanteFilante qui hérite de PlanteSimple et réécrit SimulerCroissance pour offrir la possibilité de s'étendre sur une case adjacente une seule fois .
- PlanteFilante qui hérite de PlanteSimple et réécrit SimulerCroissance et génere une probabilité d'arrêter sa croissance tant qu'elle n'est pas taillée.

Autre classes :

- Partie stocke toutes les informations nécessaire au bon fonctionnement du jeu
- Affichage stocke tous les ASCII Art et affichages lourd afin de ne pas polluer le code
- Item qui stocke les informations des Items pour les affichages
- Sauvegarde qui peut-être créer à partir de Partie ou d'un csv et qui permet de faire le lien entre les deux et de gérer les sauvegardes.
- GestionJeu qui permet de faire jouer une instance de Partie grâce à un grand nombre de méthodes. Beaucoup de méthodes représentent des pages où on peut naviguer entre elles et effectuer beaucoup d'actions puis simuler de semaine en semaine afin de faire évoluer et de gérer nos plantes et terrains.

En résumé une partie est créée grâce à la classe Partie et est simulée grâce aux méthodes de GestionJeu. Cette partie comporte toutes les informations nécessaires venant principalement des classes de Terrains qui contient une matrice de plante le Potager qui nous permet de modéliser nos données et un vrai potager. Ces Plantes et Terrains ont des

classes héritières qui permettent d'ajouter des mécaniques différentes. Et enfin cette partie peut être sauvegardée grâce à la classe Sauvegarde.

V. Tests réalisés

Nous avons réalisés 2 types de tests durant ce projet :

- Des tests individuels sur chaque classe ou fonction ajoutée nous avons créé un objet et l'avons affiché pour voir s'il se comportait comme nous le souhaitions.
- Des tests longs où nous testions le jeu sur plusieurs tours en simulant de vrais parties pour vérifier les graphismes, la fluidité, l'ergonomie et les fonctionnalités. Grâce au système de sauvegarde il était facile de tester rapidement chaque situation du Jeu.

Vous pouvez retrouver des Parties déjà avancées dans le fichier Sauvegarde afin de tester rapidement de nouvelles situation :

VI. Gestion de projet

A. Organisation de l'équipe

Notre équipe est composée de deux membres : Louis Bourely (Prépa BL) et Pierre Viven (Prépa des INP). Pour mener à bien ce projet, nous avons adopté une méthode de travail inspirée de la méthode AGILE.

Nous avons commencé par définir précisément l'idée du jeu, ses fonctionnalités et tous les objets que nous souhaitions créer. Une fois l'architecture claire et acté on se répartissait chacun des fonctions/classes à implémenter pour la prochaine session de Travail. Nous avons organisé des sessions une à deux fois par semaine pour faire régulièrement le point sur l'avancement et mettre à jour nos codes respectifs.

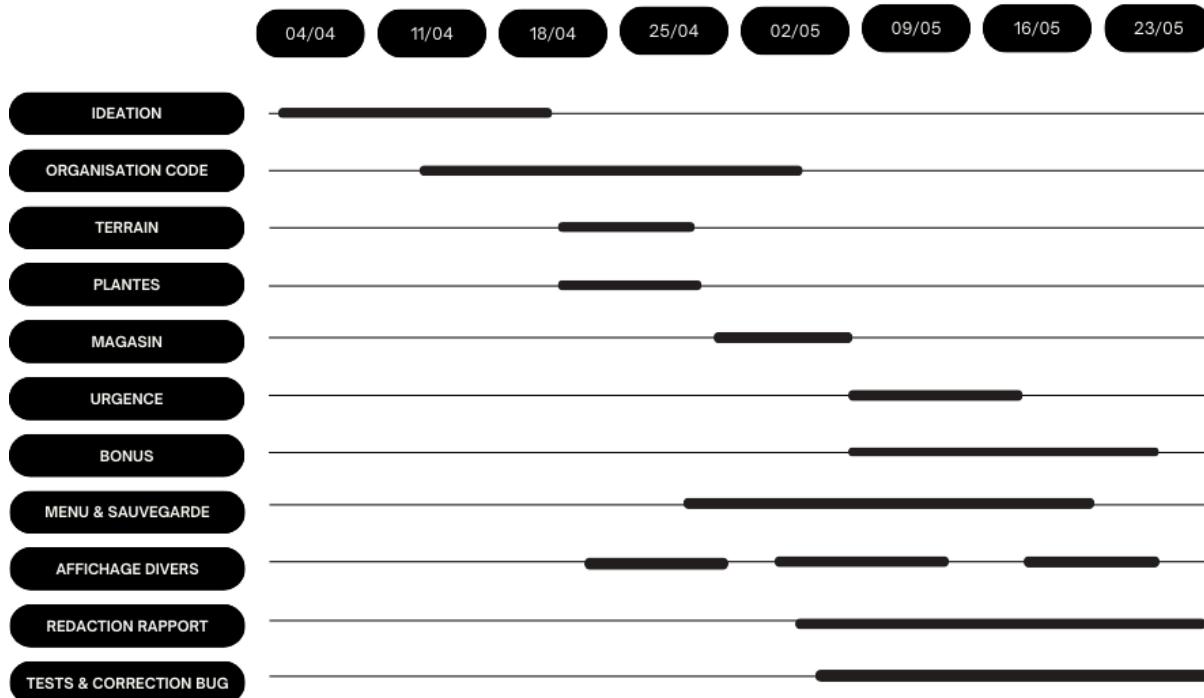
Afin d'assurer un bon suivi, nous avons mis en place un Google drive avec toute la documentation de nos fonctionnalités, les tâches à réaliser pour chacun, les consignes du projet et tous autres documents utiles. Pour le code nous avons utilisé le dépôt Github prévu.

B. Planning de réalisation

PLANNING DE PROJET

DIAGRAMME DE GANTT

ENSemenc



C. Répartition des tâches

MATRICE D'IMPLICATION : PROJET PROGAV			
Nom Projet :	Projet ENSemenc: Verdadura		date:23/05/2025
MEMBRES DE L'EQUIPE: Bourely Louis, Viven Pierre			
n°	Tâche/Fonctionnalité/Fonction	Nom du/des Codeurs	Pourcentage de participation si tâche partagée
1	Concept du jeu	Louis/Pierre	50 / 50
2	Idées Plantes/Terrains	Louis/Pierre	80 / 20

3	Idées Fonctionnalités	Louis/Pierre	40 / 60
4	Abstract Terrain	Pierre	100
5	Abstract Plante	Louis	100
6	Héritage Terrain	Louis	100
7	Héritage Plante	Louis	100
8	Accueil/GestionJeu	Pierre	100
9	Magasin	Louis/Pierre	30 / 70
10	Items	Louis	100
11	Météo	Louis	100
12	Urgences	Louis/Pierre	70 / 30
13	Sauvegarde	Pierre	100
14	GestionJeu : Terrain	Pierre	100
15	Affichages	Louis/Pierre	50 / 50
16	Informations	Louis/Pierre	50 / 50
17	Rapport	Louis/Pierre	50 / 50
REMARQUES:	<p>La création et l'idéation du réalisé en début de projet a été partagé puis fini par Louis pour la partie cractéristiques et imaginatives. Pour le code Pierre était responsable des menus du bon assemblage des classes et fonctions entres elles.</p> <p>Louis s'est occupé des Classes principales et de leur héritage.</p> <p>Chacun a suivi régulièrement les avancées de l'autre et a aidé régulièrement sur des aspects du code de l'autre.</p> <p>Le partage des tâche a été très équitable et chacun a apporté ses compétences et a pu apprendre grâce à ce projet.</p>		

VII. Bilan

Points positifs :

La dynamique et l'organisation de l'équipe ont bien fonctionné. Grâce à un travail régulier, bien partagé et une bonne entente le projet s'est bien déroulé avec une bonne attente ce qui a rendu le travail agréable et plaisant.

La création et l'idée des jeux ont été stimulantes. Nous sommes satisfait de la partie créativité qui nous a plu et que nous avons pu pousser à un bon niveau.

Les contrôles et la navigation du jeu sont un autre bon point positif, le jeu se joue facilement et rapidement.

Points à améliorer :

La logique et l'équilibrage du jeu n'ont pas été assez testé. Le jeu peut encore être amélioré notamment sur les conditions des plantes afin qu'elles aient toutes un intérêt.

Rajouter des fonctionnalités/idées, nous aurions souhaité avoir le temps de développer des maladies et des items pour les traiter ainsi que de nouvelles urgences avec des Malus et Bonus avec de nouvelles mécaniques. Nous aurions également souhaité développer un compagnon virtuel qui aide, commente et accompagne au long de l'aventure pour renforcer l'immersion et l'expérience joueur. Et en particulier donner un but final au jeu avec des succès à débloquer pour voir fixer des objectifs plus concrets. De plus nous aurions souhaité rajouter des plantes qui se vendent avec un système de bourse avec un prix qui augmente et baisse pour favoriser des stratégies de jeu. Enfin nous aurions aimé rajouter des niveaux de difficulté pour faire varier les paramètres initiaux mais aussi les probabilités d'événements.

L'optimisation du code pourrait être plus poussée. Certains affichages ASCII ou codes peuvent être amélioré ou rassembler avec plus de temps

En conclusion, ce projet nous a permis de mettre en application nos compétences sur la programmation objet. Ce projet nous a permis de manipuler concrètement ces concepts et de monter en compétence dessus. C'était aussi un challenge technique et créatif stimulant qui nous a plu. C'est une expérience enrichissante pour de futurs projets. En espérant que le Jeu vous plaira.