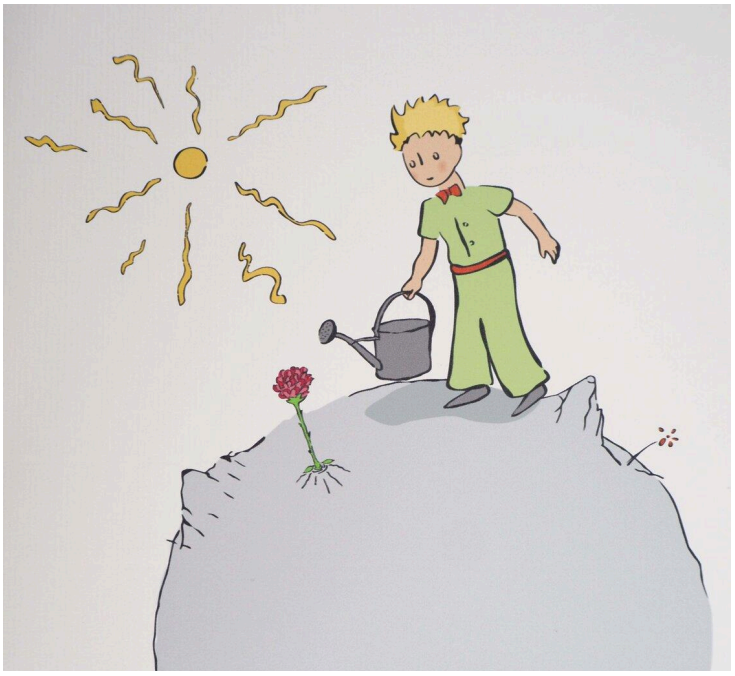
BORDEAUX INP Ensc

Programmation avancée - Projet 2024

ENSemenCe



Gautier Adélie - Prudhommeaux Erwan

Sommaire

I - Introduction

L'objectif de ce projet est de programmer un jeu de simulation de potager, permettant de créer et cultiver son propre jardin. Le jeu devait intégrer de nombreux paramètres impactants la croissance des plantes, afin de proposer une expérience de jeu riche.

Le jeu est codé en C#, s'affiche grâce à la console, et utilise les principes de programmation objet que nous avons vu en cours.

Nous étions totalement libres sur le choix du pays (imaginaire ou non) dans lequel se trouverait le potager, et des types de plantes disponibles. De notre côté, nous avons décidé de nous inspirer de l'univers de Petit Prince, issu de l'œuvre de Saint-Exupéry, pour créer notre jardin.

II - Description du jeu

1 - Présentation de l'univers

Le jeu se déroulant dans l'univers de Saint-Exupéry, le premier terrain disponible pour faire son potager correspond à la planète du Petit Prince. Il s'agit du terrain de base, permettant de faire pousser les premières plantes du jeu. En début de partie, on peut cultiver des étoiles, des météorites, des roses, des chapeaux et des nuages.

Chaque plante possède des caractéristiques différentes, en termes de vitesse de croissance, de nombre de fruits donnés, de longévité… Il y a deux grands types de plantes : les monocarpiques, qui meurent une fois qu'on a récolté les fruits (il faut alors les déraciner pour planter une nouvelle graine) et les polycarpiques, qui peuvent faire plusieurs tournées de fruits avant de disparaître.

De plus, pour bien pousser, les plantes doivent avoir leurs besoins de complétés. Il existe les besoins classiques en eau et lumière, mais nous en avons également ajouté un troisième type : l'applaudissement. En effet un certain type de plante en particulier, la plante orgueilleuse, a besoin d'un nombre suffisant d'applaudissements pour pousser correctement.

Au cours de l'avancée de la partie, le joueur peut acheter de nouveaux terrains pour agrandir son potager. Les différents terrains qui se débloquent correspondent chacun à la planète d'un personnage différent : le Businessman, le Buveur, le Vaniteux, le Roi, le Géographe et l'Allumeur de réverbères. Chaque terrain possède ses caractéristiques particulières. Des nouvelles plantes sont débloquées avec les terrains, certaines plantes poussent mieux suivant la planète, et les terrains apportent aussi leur propres catastrophes.

Les plantes qui arrivent plus tard dans la partie sont donc les étoiles filantes (des plantes qui se déplacent par elles-même et laissent de la poussière d'étoile derrière elles), les alcootier (des arbres produisant de l'alcool, qui prennent 4 cases de large), les plantes orgueilleuses, les couronnes, les planètes et les lampadaires. Les lampadaires sont un type très particulier. Il ne font pas de fruit

(ça ne sert donc à rien de les récolter), mais produisent de la lumière sur les 9 cases autour d'eux. Cela permet donc de compléter le besoin en lumière des plantes environnantes.

Notre jardin peut également être envahit par les mauvaises herbes, qui prennent toute la place sans pouvoir être récoltées. On retrouve deux types de mauvaises herbes : les baobabs, mettant du temps à pousser et prenant 4 places de large, et les champignons, plus rapides et plus petits.

Des animaux peuvent aussi venir visiter le jardin. Parmi eux se trouvent deux nuisibles. D'abord les éléphants, qui avancent en ligne droite et écrasent tout sur leur passage. Toutes les plantes écrasées disparaissent, même les arbres tels que les baobabs. Ensuite, les oiseaux, qui apparaissent par groupe et picorent les plantes jusqu'à ce qu'elles disparaissent. A chaque tour, une plante picorée perd quelques points de vie.
La faune contient aussi deux animaux amicaux. Les moutons se dirigent vers les baobabs pour les manger, débarrassant ainsi le jardin des mauvaises herbes. Les serpents mangent les éléphants, les faisant disparaître immédiatement. De plus, ces derniers peuvent se cacher sous les chapeaux. En effet, si un serpent se trouve sur la même case qu'une plante chapeau, il aura le statut caché et ne sera alors pas impacté par certaines actions du joueur (tel que "Effrayer" par exemple).

Dans notre univers, il existe trois types de saisons différentes (saison calme, saison de sécheresse et saison de gel) et plusieurs météos possibles (soleil, nuit, pluie, calme). Parfois, des catastrophes peuvent se déclencher.

Enfin, les plantes peuvent attraper des maladies. Il existe différentes maladies, qui possèdent des caractéristiques d'apparition et de propagation diverses. Ces maladies n'entraînent pas non plus toutes les mêmes symptômes : certaines ont un impact directement sur la vie des plantes, tandis que d'autres vont jouer sur leur niveau de besoin (en eau par exemple).

Dans notre univers, nous avons également un système de marché, où l'on peut acheter et vendre des produits. L'argent se compte alors en poussière d'étoiles.

2 - Déroulement d'une partie

La première étape lorsqu'un joueur commence une partie est de placer son premier terrain, celui du Petit Prince. Il peut le placer à l'endroit où il le souhaite sur une matrice de 3x3. Cette matrice se complètera au fur et à mesure de la partie, en achetant des nouveaux terrains qui seront positionnables là où on le souhaite dans les cases disponibles. Chaque terrain fait une taille de 7*7 cases.

Ensuite, le joueur a le choix entre trois actions : entretenir son jardin, aller au magasin ou finir la journée.

S'il choisit d'entretenir son jardin, il a accès à de nombreuses actions possibles. Il peut planter une graine, arroser, récolter ou protéger une plante, déraciner une plante ou une mauvaise herbe, applaudir ou effrayer. Effrayer permet de faire disparaître tous les animaux du jardin, tandis

qu'applaudir complète les besoins en applaudissements des plantes et fait reculer d'une case tous les animaux présents autour. Ces actions se réalisent en choisissant sur notre terrain la case sur laquelle faire l'action, puis en validant le choix de ce qu'il veut faire.

Si il ouvre le magasin, le joueur peut acheter un terrain, des graines ou encore des objets permettant de faire les différentes actions décrites précédemment (une pelle pour déraciner, une lanterne pour éclairer, un haut parleur pour effrayer…). Il peut également vendre tout ce qu'il a sur lui (les objets comme les fruits récoltés des plantes).

Lorsque l'on finit une journée, on termine le tour de jeu. 7 jours vont alors passer, et nous retrouvons notre potager qui a évolué. C'est alors le début d'une nouvelle semaine, et d'un nouveau tour de jeu. Le joueur peut alors répéter les actions décrites précédemment.

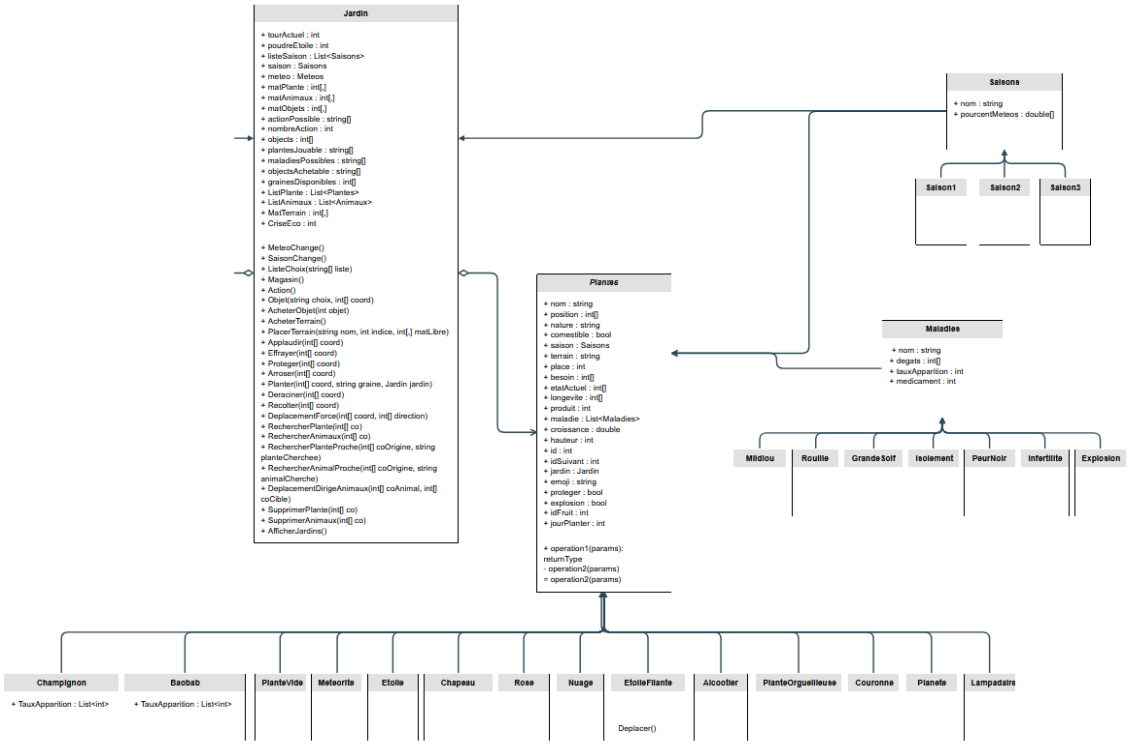
Chaque journée se termine automatiquement lorsque l'on ne possède plus d'actions. Le joueur peut réaliser 3 actions maximum à chaque tour. De plus, il y a une petite chance qu'une catastrophe arrive lors d'une journée. A ce moment-là, le jeu passe en temps réel pendant les 5 minutes de la catastrophe. Le système d'action du joueur reste le même mais le potager évolue constamment pendant 5 minutes.

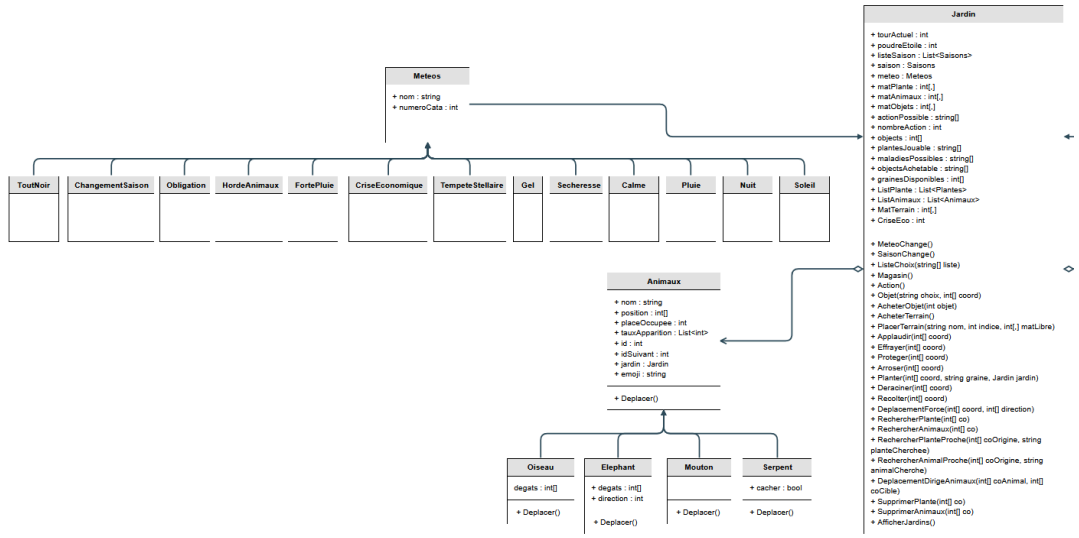
III - Explication du code

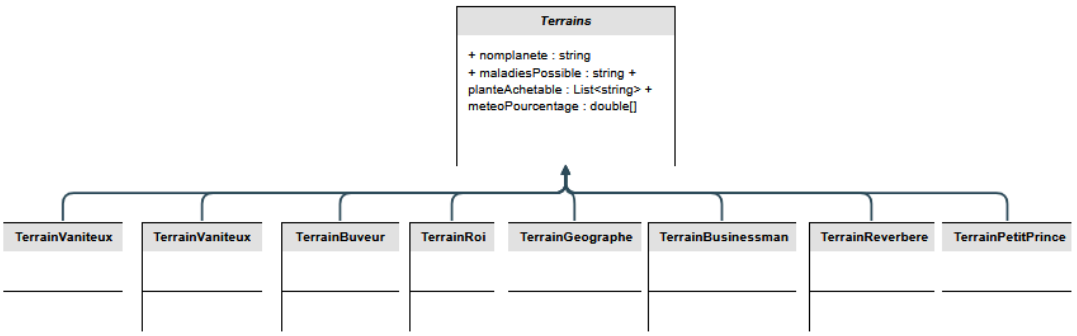
1 - Modélisation objet

Pour programmer ce jeu, nous avons principalement utilisé le principe de la programmation orientée objet. Nous avons créé de nombreuses classes pour gérer les différents éléments présents dans et autour du jardin. Nous avons une classe principale, la classe Jardin, qui s'occupe de gérer tous les évènements qui se passent dans le jardin. C'est celle-ci qui contient les matrices modélisant le plateau de jeu (une matrice pour enregistrer la position des plantes et une seconde pour la position des animaux), les listes de plantes qui ont été ajoutées au jardin, les objets possédés par le joueur… Elle contient également toutes les méthodes permettant au joueur de faire ses actions, et de quoi gérer les météos et saisons. Cette classe Jardin fait appel aux différentes classes et sous classes que nous avons défini pour chaque élément.

Nous avons tracé les diagrammes UML pour décrire au mieux les classes et les relations entre elles :


Jardin
+ tourActuel : int
+ poudreEtoile : int
+ listeSaison : List<Saisons>
+ saison : Saisons
+ meteo : Meteos
+ matPlante : int[,]
+ matAnimaux : int[,]
+ matObjets : int[,]
+ actionPossible : string[]
+ nombreAction : int
+ objects : int[]
+ plantesJouable : string[]
+ maladiesPossibles : string[]
+ objectsAchetable : string[]
+ grainesDisponibles : int[]
+ ListPlante : List<Plantes>
+ ListAnimaux : List<Animaux>
+ MatTerrain : int[,]
+ CriseEco : int
+ MeteoChange()
+ SaisonChange()
+ ListeChoix(string[] liste)
+ Magasin()
+ Action()
+ Objet(string choix, int[] coord)
+ AcheterObjet(int objet)
+ AcheterTerrain()
+ PlacerTerrain(string nom, int indice, int[,] matLibre)
+ Applaudir(int[] coord)
+ Effrayer(int[] coord)
+ Proteger(int[] coord)
+ Arroser(int[] coord)
+ Planter(int[] coord, string graine, Jardin jardin)
+ Deraciner(int[] coord)
+ Recolter(int[] coord)
+ DeplacementForce(int[] coord, int[] direction)
+ RechercherPlante(int[] co)
+ RechercherAnimaux(int[] co)
+ RechercherPlanteProche(int[] coOrigine, string planteCherchee)
+ RechercherAnimalProche(int[] coOrigine, string animalCherche)
+ DeplacementDirigeAnimaux(int[] coAnimal, int[] coCible)
+ SupprimerPlante(int[] co)
+ SupprimerAnimaux(int[] co)
+ AfficherJardins()
Saisons
+ nom : string
+ pourcentMeteos : double[]
Saison1
Saison2
Saison3
Plantes
+ nom : string
+ position : int[]
+ nature : string
+ comestible : bool
+ saison : Saisons
+ terrain : string
+ place : int
+ besoin : int[]
+ etatActuel : int[]
+ longevite : int[]
+ produit : int
+ maladie : List<Maladies>
+ croissance : double
+ hauteur : int
+ id : int
+ idSuivant : int
+ jardin : Jardin
+ emoji : string
+ proteger : bool
+ explosion : bool
+ idFruit : int
+ jourPlanter : int
+ operation1(params): returnType
- operation2(params)
= operation2(params)
Maladies
+ nom : string
+ degats : int[]
+ tauxApparition : int
+ medicament : int
Mildiou
Rouille
GrandeSoif
Isolement
PeurNoir
Infertilite
Explosion
Champignon
+ TauxApparition : List<int>
Baobab
+ TauxApparition : List<int>
PlanteVide
Meteorite
Etoile
Chapeau
Rose
Nuage
EtoileFilante
Deplacer()
Alcootier
PlanteOrgueilleuse
Couronne
Planete
Lampadaire


Meteos
+ nom : string
+ numeroCata : int
ToutNoir
ChangementSaison
Obligation
HordeAnimaux
FortePluie
CriseEconomique
TempeteStellaire
Gel
Secheresse
Calme
Pluie
Nuit
Soleil
Animaux
+ nom : string
+ position : int[]
+ placeOccupee : int
+ tauxApparition : List<int>
+ id : int
+ idSuivant : int
+ jardin : Jardin
+ emoji : string
+ Deplacer()
Oiseau
degats : int[]
+ Deplacer()
Elephant
+ degats : int[]
+ direction : int
+ Deplacer()
Mouton
+ Deplacer()
Serpent
+ cacher : bool
+ Deplacer()
Jardin
+ tourActuel : int
+ poudreEtoile : int
+ listeSaison : List<Saisons>
+ saison : Saisons
+ meteo : Meteos
+ matPlante : int[,]
+ matAnimaux : int[,]
+ matObjets : int[,]
+ actionPossible : string[]
+ nombreAction : int
+ objects : int[]
+ plantesJouable : string[]
+ maladiesPossibles : string[]
+ objectsAchetable : string[]
+ grainesDisponibles : int[]
+ ListPlante : List<Plantes>
+ ListAnimaux : List<Animaux>
+ MatTerrain : int[,]
+ CriseEco : int
+ MeteoChange()
+ SaisonChange()
+ ListeChoix(string[] liste)
+ Magasin()
+ Action()
+ Objet(string choix, int[] coord)
+ AcheterObjet(int objet)
+ AcheterTerrain()
+ PlacerTerrain(string nom, int indice, int[,] matLibre)
+ Applaudir(int[] coord)
+ Effrayer(int[] coord)
+ Proteger(int[] coord)
+ Arroser(int[] coord)
+ Planter(int[] coord, string graine, Jardin jardin)
+ Deraciner(int[] coord)
+ Recolter(int[] coord)
+ DeplacementForce(int[] coord, int[] direction)
+ RechercherPlante(int[] co)
+ RechercherAnimaux(int[] co)
+ RechercherPlanteProche(int[] coOrigine, string planteCherchee)
+ RechercherAnimalProche(int[] coOrigine, string animalCherche)
+ DeplacementDirigeAnimaux(int[] coAnimal, int[] coCible)
+ SupprimerPlante(int[] co)
+ SupprimerAnimaux(int[] co)
+ AfficherJardins()



2 - Mode urgence

Lorsque le mode urgence est activé, le jeu passe en temps réel. Le joueur peut voir l'heure défiler au-dessus du jardin, et le plateau évoluera en même temps qu'il fait ses actions. (contrairement au mode normal où le plateau s'actualise à chaque fois que le joueur a terminé une action). Ce mode durera 5 minutes. Pour que notre jeu puisse fonctionner en faisant plusieurs actions en même temps, nous avons utilisé des Thread, issus de la méthode System.Threading. Ils nous ont servi pour avoir une bien meilleure cohésion. En effet, ils permettent de créer un objet qui va exécuter un programme sans s'arrêter.

IV - Gestion de projet

1 - Organisation du travail

Étant par groupe de deux pour réaliser ce projet, nous avons dû nous organiser et répartir le travail. Nous avons utilisé la plateforme Github pour nous partager le code, afin de pouvoir travailler chacun de notre côté avant de mettre facilement le code en commun.

La première étape du projet a été de choisir l'univers dans lequel allait se trouver le potager, et définir l'ensemble des fonctionnalités que nous aimerions ajouter au jeu. Cette étape de réflexion, avant même de commencer à coder, est une partie importante. Elle nous a pris beaucoup de temps, environ deux des séances de TD dédiées au projet, mais il s'agissait d'une étape primordiale car elle impacterait toute la structure de notre projet par la suite.
En plus de décider de tout le fonctionnement du jeu, nous nous sommes occupés de définir la structure du code ; les classes que nous allions utiliser, leurs attributs principaux, le nombre de sous classe, ce qu'elles feraient... Nous avons également créé toutes les plantes et noté les paramètres de chacune d'entre elles (vitesse de croissance, terrain préféré, type…).

Une fois que cela était réglé, nous nous sommes répartis le travail pour commencer la programmation. Nous nous sommes chacun occupés de classes différentes, afin de pouvoir travailler en parallèle sans avoir besoin de ce que faisait l'autre. Lorsque nous devions faire appel à une classe

qui n'avait pas encore été créée nous pouvions déjà le noter, car nous savions comment cette classe fonctionnerait grâce au travail fait en amont.

A la fin du projet, l'un de nous s'est occupé de rassembler toutes les fonctionnalités du code pour avoir un jardin fonctionnel. Cela marche avec une fonction dans le programme principal, qui fait tourner la simulation en appelant chacune des classes codées précédemment ainsi que les différentes méthodes, pour réaliser un tour de jeu. Pendant ce temps-là, le deuxième s'est chargé de commencer la rédaction du rapport.

2 - Test

Tout au long du projet, nous avons régulièrement fait des tests pour vérifier que notre code fonctionnait correctement. Au début du projet nous avons commencé par créer des classes avant de faire la fonction de simulation qui allait gérer l'ensemble du programme. Nous ne pouvions donc pas l'utiliser pour faire les tests de tout le jardin, mais nous pouvions créer des instances séparées pour tester les différentes classes. Cela permet de pouvoir tester toutes les méthodes d'une classe au cours de la programmation, bien que nous ne pouvions pas toujours essayer les liens entre différentes classes.

Une fois le jeu terminé, nous avons pu le tester dans son ensemble, avec tous les liens et toutes les fonctionnalités. Pour cela, nous avons réalisé des parties en testant les différentes actions possibles. Nous avons tenté de faire des actions impossibles, afin de voir si les erreurs étaient bien prises en compte par notre programme pour ne pas générer de bug.

De plus, lorsque le jeu était fonctionnel, nous avions tout un travail de rééquilibrage à faire. En effet, lorsque nous avons établi les propriétés de nos plantes au tout début du projet, nous avons décidé des différents paramètres (quantités, vitesses, besoins…) arbitrairement. Nous n'avions aucune idée de si ces paramètres étaient cohérents entre eux, et si le jeu ne serait pas trop facile ou trop dur suivant les cas. Nous avons donc dû jouer des parties, afin de tester les différents paramètres et de les modifier pour que le jeu soit le plus équilibré possible.

V - Conclusion

Ce projet était très intéressant à réaliser. Il nous a permis de réellement mettre en application ce que nous avions vu pendant les cours. Il s'agissait d'un projet complexe, avec des classes conséquentes, de nombreuses fonctionnalités, et beaucoup de relations entre les classes. Chercher à faire tout cela nous a aidé à vraiment comprendre le fonctionnement de la programmation orientée objet. En effet, nous n'avions pas forcément bien compris toutes les subtilités de ces méthodes lors des cours en CM, et les TD mettaient en oeuvres des exercices moins complexes et donc utilisant surtout les principes de base. Nous confronter ici à de tels problèmes nous a poussé à regarder en détail la POO.

De plus, le fait d'être extrêmement libres dans le choix du thème est une chose que nous avons grandement appréciée. Nous avons pu nous amuser à inventer tout un univers, avec des plantes et des façons de fonctionner, ou encore des mécaniques particulières et originales, par exemple sur les maladies ou les catastrophes possibles. Il s'agissait d'une partie amusante que de faire appel à notre imagination pour créer tout cela. De plus, nous avons pu nous adapter pour faire vraiment quelque chose qui nous plaisait, et non pas quelque chose d'imposé par une consigne.

Cependant, nous avons rencontré quelques difficultés au cours du projet. La principale limite que nous avons eu est par rapport au temps que nous avions, et à la quantité de travail. Celle-ci est un peu liée au point précédent. En effet, le fait d'être aussi libre dans le travail nous a donné envie de faire beaucoup de choses et d'ajouter un grand nombre de fonctionnalités. Mais nous nous sommes rendu compte plus tard que nous avions vu trop grand dans nos objectifs, et que nous ne pourrions pas terminer avec le temps que nous avions. Nous avons donc été obligés de simplifier voir supprimer plusieurs de nos idées. En plus de cela, nous n'avons pas eu le temps de corriger tous les problèmes présents au sein du code. En effet, le jeu fonctionne et se lance correctement, mais il y a parfois certaines fonctionnalités qui ne donnent pas les résultats attendus. Cela est très frustrant, d'autant plus qu'il ne s'agit pas d'une question de difficulté. Avec plus de temps, nous aurions pu tout implémenter, et régler tous les petits soucis. Nous avons donc été obligés de laisser dans le code certains débuts de fonctionnalités en précisant qu'ils n'ont pas encore été implémentés.

De plus, avec du temps supplémentaire, nous aurions également aimé regarder plus attentivement la partie rééquilibrage. En effet, nous n'avons pu faire que très peu de parties pour tenter d'équilibrer les différents paramètres du jeu, et nous savons que la difficulté de celui-ci n'est pas très bien réglée.

Annexes

Matrice d'implication

MATRICE D'IMPLICATION : PROJET PROGAV		
Nom projet :	**projet-ensemenc-Gautier-Adelie-Prudhommeaux-Erwan**	**date:**
MEMBRES DE L'EQUIPE : Erwan Prudhommeaux, Adélie Gautier		

n°	Tâche/Fonctionnalité/Fonction	Nom du/des Codeurs	Pourcentage de participation si tâche partagée
1	Réflexion sur l'univers choisi et l'ensemble des fonctionnalités à implanter	Erwan / Adélie	50 / 50
2	classe Animaux et sous classes dérivées	Adélie	/
3	Classe Maladie et sous classes dérivées	Adélie	/
4	Classe Plantes et sous classes dérivées	Adélie	/
5	Classe Terrains et sous classes dérivées	Erwan	/
6	Implémentation des météos	Erwan	/
7	Implémentation des saisons	Erwan	/

8	Magasin	Erwan	/
9	Gestion des terrains	Erwan	/
10	Actions	Erwan / Adélie	50 / 50
11	Affichage jardin	Erwan	/
12	Gestion des plantes et animaux du jardin	Adélie	/
13	Program.cs	Erwan	/
14	Tests et équilibrage	Erwan / Adélie	90 / 10
15	Diagramme UML	Adélie	/
16	Rédaction du rapport	Erwan / Adélie	10 / 90