GALLOU Antoine

SAINTE-CROIX Lucas

PROJET DE PROGRAMMATION AVANCÉE

Rapport Technique

ENSEMENC VS MARIO

23 mai 2025

SOMMAIRE

I. Univers du jeu et spécificités

Le jeu que nous avons développé propose au joueur d'incarner un personnage fictif, Toad, dans le but de faire prospérer un jardin en cultivant diverses plantes tout en faisant face à des imprévus, notamment les interventions perturbatrices de Mario en kart. La simulation alterne entre deux modes de jeu : le mode classique et le mode urgence. Chaque semaine dans le jeu correspond à un tour, durant lequel l'état des plantes évolue en fonction des conditions climatiques et des actions du joueur.
Nous allons maintenant parler plus en détail de notre univers.

II. Possibilités des joueurs et déroulement d'une partie

Mode Classique

Le mode classique constitue la phase principale et régulière du jeu. Lors du lancement de la partie, le joueur sélectionne une saison de départ parmi les quatre proposées : hiver, printemps, été ou automne. Cette saison détermine les conditions climatiques initiales (quantité de pluie, ensoleillement, température) qui influencent directement la croissance des plantes. Ces données sont recalculées chaque semaine via l'instance de saison en cours.

À chaque tour, les plantes présentes sur la grille sont simulées une par une. Si une maladie est localisée sur la même case qu'une plante, celle-ci peut être infectée – à condition qu'elle fasse partie des cibles privilégiées de cette maladie. Certaines affections, comme les Toxinelles, provoquent une dégradation accélérée de la santé de la plante. Si la plante survit et que ses conditions de croissance sont réunies (bonne exposition, espace suffisant, climat adéquat), elle poursuit sa croissance, pouvant atteindre la maturité et produire des fruits.

Le joueur peut intervenir dans son jardin grâce à un menu d'actions complet :

- Arroser une plante (réduction temporaire de son besoin en eau),
- Cueillir les fruits (à condition que la plante soit mature et fructifère),
- Semer une graine (parmi quatre espèces : Fleur de feu, Fleur de glace, Arbre à cheddar, Plante carnivore),
- Utiliser un objet spécial,
- Déraciner une plante (pour libérer l'espace d'une case),
- Consulter un diagnostic d'état pour une plante spécifique,
- Obtenir des fiches descriptives sur chaque espèce végétale.

Parmi les objets spéciaux disponibles dans l'inventaire, plusieurs ont des effets particuliers :

- Etoiles : augmentent la satisfaction d'une plante ciblée,
- Boules de feu et boules de glace : prévus pour influencer les terrains spécifiques ou contrer des maladies, mais leurs effets restent à implémenter dans la version actuelle,
- Sérums : soignent les plantes infectées,
- Cheddars : boostent les Plantes Carnivores, notamment pour accélérer leur action de nettoyage face aux Toxinelles.

À la fin de chaque tour, le joueur peut choisir de passer à la semaine suivante, ce qui entraîne une mise à jour globale de l'état du jardin (croissance, maladie, météo, etc.).

Mode Urgence

Le mode urgence est un événement récurrent qui se déclenche automatiquement toutes les cinq semaines. Lorsqu'il survient, Mario fait une apparition spectaculaire à bord de son kart, menaçant de détruire une ligne entière de la grille.

Le joueur reçoit alors une alerte. Il peut tenter de neutraliser Mario à l'aide d'un éclair, si son inventaire en contient. En cas de succès, le kart est foudroyé et les plantes sont épargnées. Dans le cas contraire, une ligne complète de plantation est irrémédiablement écrasée.

Ce mode ne se contente pas de représenter une menace ponctuelle : à la suite du passage du kart, quatre maladies apparaissent aléatoirement sur la grille. Chaque maladie génère une nouvelle zone de danger, pouvant infecter les plantes situées dans un rayon de 1 autour de sa position.

À la fin de ce mode, le jeu revient automatiquement en mode classique et la simulation reprend son cours normal avec l'avancée d'une semaine supplémentaire.

III. Modélisation objet réalisée

Description des principales classes

Calendrier

La classe *Calendrier* joue un rôle central dans la gestion du temps au sein du jeu. Elle modélise le suivi des semaines, représentant ainsi la progression temporelle au fil de l'année de la simulation.

Chaque instance de cette classe conserve le numéro de la semaine en cours, un entier compris entre 1 et 52, ce qui permet de segmenter l'année en quatre saisons distinctes.

Grâce à la propriété en lecture seule *saisonCourante*, la classe détermine la saison actuelle en fonction de la semaine. Cette propriété renvoie une instance correspondant à l'une des quatre saisons — hiver, printemps, été ou automne — selon un découpage fixe en trimestres de 13 semaines. Ce découpage facilite la gestion des conditions climatiques et de l'environnement des plantes, qui évoluent selon la saison.

La méthode *AvancerSemaine* permet de faire progresser le calendrier d'une semaine. Lorsque le compteur dépasse la semaine 52, il revient à 1, simulant ainsi le passage à une nouvelle année dans le jeu. Cette fonctionnalité est essentielle pour rythmer les événements, les cycles de croissance des plantes et les différentes phases du gameplay.

Case

La classe *Case* représente une unité élémentaire de la grille de jeu, matérialisant une position précise définie par ses coordonnées *x* et *y*. Chaque case peut contenir un terrain spécifique et éventuellement une plante, bien que la présence d'une plante ne soit pas obligatoire, ce qui est reflété par la possibilité pour l'attribut *plante* d'être nul.

La méthode *AfficherEmoji* joue un rôle important dans la visualisation du jeu : elle permet de déterminer quel symbole doit être affiché pour représenter le contenu de la case dans l'interface. Si la case est vide, un espace vide est renvoyé. Si une plante y est présente mais qu'elle est morte, un « X » est affiché, indiquant son état. Sinon, chaque type de plante est associé à une lettre spécifique (par exemple « F » pour *FleurDeFeu* ou « G » pour *FleurDeGlace*), facilitant la reconnaissance visuelle.

La méthode *GetCouleurBordure* définit la couleur des bordures entourant la case en fonction du type de terrain qu'elle contient. Cette personnalisation visuelle aide à différencier les environnements (fromage, océan, volcan, arc-en-ciel) et apporte une richesse graphique à la grille.

Ainsi, la classe *Case* constitue la base de la représentation spatiale dans le jeu, en associant à la fois position, environnement et contenu vivant.

Grille

La classe *Grille* constitue la structure spatiale principale du jeu : elle représente le jardin que le joueur devra cultiver et protéger. Elle est construite comme une matrice de dimensions 6x6, contenant des objets de type *Case*. Chaque *Case* peut accueillir une plante et est associée à un type de *Terrain* spécifique, ce qui donne au jardin une géographie visuellement distincte et ludique.

La méthode de construction de la grille répartit les terrains de manière fixe et symétrique : les différentes zones (Fromage, Océan, Volcan, Arc-en-ciel) sont attribuées à des quadrants prédéfinis de la grille, renforçant la lisibilité de l'interface et l'identité des environnements.

L'affichage de la grille est particulièrement travaillé. À travers les méthodes *AfficherLigneSeparatrice* et *AfficherLigneContenu*, la grille est imprimée dans la console avec un système de bordures colorées en fonction du terrain, et des symboles qui représentent l'état des plantes. Cette visualisation dynamique rend l'expérience de jeu plus engageante et permet une compréhension immédiate de la situation.

La méthode *PlacerPlante* permet au joueur d'ajouter une plante à une case spécifique, en s'assurant que les coordonnées soient valides. La grille conserve ainsi l'intégrité de ses données spatiales.

Enfin, la méthode *RecupererVoisines* joue un rôle important pour les mécaniques de propagation (comme les maladies ou les effets de certaines plantes), en identifiant les cases voisines d'un point donné. Cela permet d'implémenter des logiques d'interaction locale dans le jeu.

En somme, *Grille* est à la fois un support visuel, une carte de terrain et un outil de gestion spatiale indispensable au bon fonctionnement de la simulation.

Inventaire

La classe *Inventaire* représente l'ensemble des ressources disponibles pour le joueur au cours de la partie. Ces ressources sont utilisées pour influencer directement le déroulement du jeu, en permettant des actions spéciales ou des interactions ciblées avec les plantes ou les éléments perturbateurs du jardin.

Chaque type de ressource possède une fonction bien définie dans les mécaniques de jeu :

- nbEclairs : ces objets permettent de stopper Mario lorsqu'il intervient en mode Urgence, évitant ainsi des perturbations majeures dans le jardin.
- nbSerums : ils servent à soigner les maladies contractées par les plantes. Leur utilisation est cruciale pour assurer la survie et la croissance des végétaux infectés.
- nbEtoiles : elles boostent directement le taux de satisfaction d'une plante, le plaçant au maximum. Cela garantit non seulement sa survie, mais accélère également sa croissance.
- nbBoulesFeu et nbBoulesGlace : bien que leur fonctionnalité ne soit pas encore implémentée, leur présence anticipe des mécaniques futures. Elles sont

respectivement prévues pour contrer des maladies spécifiques et modifier les propriétés saisonnières des terrains (comme dégeler un terrain Océan en hiver ou solidifier un terrain Fromage en été).
- nbCheddars : cette ressource est destinée à interagir avec une Plante Carnivore, introduisant ainsi une forme d'alimentation ou d'entretien propre à certaines plantes.

Le constructeur de la classe initialise l'inventaire avec des quantités nulles (ou presque nulles), ce qui impose au joueur de gagner ou de gérer stratégiquement ses ressources pour faire face aux événements du jeu. Cette logique favorise l'aspect progression et planification.

En résumé, *Inventaire* est un outil de gestion essentiel qui relie directement les actions du joueur à l'état du jardin. Il offre un levier stratégique important, ajoutant de la profondeur à la simulation en introduisant des choix tactiques et des éléments de rareté.

Kart

La classe *Kart* incarne l'un des éléments perturbateurs majeurs du jeu : les attaques surprises de Mario en kart. Elle est spécifiquement mobilisée lors du passage en mode Urgence, introduisant un risque soudain qui menace directement les cultures du joueur. Cette mécanique vient renforcer l'aspect dynamique du jeu en ajoutant des événements aléatoires et potentiellement destructeurs.

Trois attributs structurent cette classe :

- ligneCible : correspond à la ligne de la grille que Mario va tenter de traverser. Elle est choisie aléatoirement, ce qui renforce l'imprévisibilité de l'événement.
- frappéParEclair : indicateur booléen qui permet de savoir si le joueur a réussi à arrêter Mario à temps à l'aide d'un éclair issu de son inventaire.
- rng : générateur de nombres aléatoires utilisé pour désigner aléatoirement la ligne cible.

Les principales méthodes définissent les différentes phases d'intervention du kart :

- Rouler(int maxLignes) : sélectionne au hasard une ligne de la grille et en avertit le joueur, instaurant ainsi une tension dramatique.
- FoudroyerKart() : permet d'interrompre l'attaque si le joueur dispose d'un éclair. Ce geste stratégique bloque l'effet destructeur du kart pour le tour en cours.
- DetruireLigne(Grille grille) : inflige les conséquences si le kart n'a pas été arrêté. Toutes les plantes sur la ligne cible sont détruites (leur état est alors défini comme mort). Cette méthode applique donc concrètement l'effet négatif de l'événement.

La conception de cette classe introduit un véritable facteur de risque dans la simulation. Elle oblige le joueur à anticiper, à stocker certaines ressources (notamment les éclairs), et à composer avec une part de hasard qui vient pimenter l'expérience de jeu. Cette instabilité rend la gestion du jardin plus exigeante et engageante.

Maladie

La classe abstraite *Maladie* introduit un autre niveau de difficulté dans la gestion du jardin : celui des affections biologiques. Elle représente un type de menace indirecte mais persistante, affectant certaines plantes de manière ciblée. L'apparition de maladies pousse le joueur à surveiller régulièrement l'état de ses cultures et à utiliser des objets spécifiques pour les soigner à temps.

Chaque maladie est localisée par ses coordonnées *(x, y)* et possède une faiblesse, c'est-à-dire un élément du jeu (souvent un objet de l'inventaire) qui permet d'y remédier. La classe garde aussi en mémoire le type de plante qu'elle peut infecter, ainsi que la gravité de l'infection, représentée par une réduction du taux de satisfaction de la plante touchée. Cette gravité est fixée à 0.3, ce qui signifie qu'une plante infectée verra son bien-être considérablement chuter si elle n'est pas soignée rapidement.

Deux méthodes essentielles sont définies ici. *EstPlanteCible* permet de vérifier si une plante est compatible avec la maladie — autrement dit, si elle peut être infectée. *Infecter* applique la maladie à la plante, à condition que celle-ci ne soit pas déjà malade, et affiche un message pour prévenir le joueur de l'infection.

En étant abstraite, cette classe sert de modèle aux différentes maladies spécifiques implémentées dans le jeu (comme la glaglaose ou la quécalorose). Cela permet d'enrichir facilement le système biologique du jeu, tout en centralisant les comportements communs des maladies.

En somme, la classe *Maladie* apporte un enjeu de prévention et de réaction dans la gestion du jardin. Elle renforce l'idée que faire prospérer un écosystème demande non seulement de bonnes conditions de croissance, mais aussi de la vigilance face aux menaces invisibles.

Plante

La classe abstraite *Plante* constitue le cœur biologique du jeu : elle modélise les végétaux que le joueur peut cultiver, surveiller et faire prospérer au fil des saisons. En tant que classe mère, elle sert de base commune aux différentes espèces de plantes présentes dans le jeu (fleurs de feu, arbres à cheddar, etc.), et regroupe à ce titre un grand nombre de propriétés et de comportements.

Chaque plante est définie par un ensemble riche d'attributs qui traduisent ses besoins, sa physiologie et ses préférences.

On y retrouve des données basiques comme sa position dans la grille *(x, y)*, son nom ou encore son espérance de vie, mais aussi des éléments plus fins comme la saison idéale pour la planter, le type de terrain qu'elle préfère, les quantités d'eau et de lumière nécessaires à son épanouissement, ou encore les plages de température dans lesquelles elle peut survivre. Elle possède également des attributs dynamiques : par exemple, *semaineDepuisSemis* reflète son âge, *boostSatisfaction* indique si elle bénéficie d'un bonus temporaire (obtenu via une étoile), et *maladie* référence une éventuelle infection en cours.

Le cœur fonctionnel de cette classe repose sur plusieurs méthodes. *Satisfaction* calcule un score de bien-être global de la plante, basé sur ses conditions de culture. Ce score détermine sa croissance via la méthode *Croissance*, qui simule la progression d'une plante vers sa maturité. La croissance est ralentie ou accélérée selon le niveau de satisfaction, et la plante peut tomber malade ou mourir si les conditions sont mauvaises ou si elle dépasse son espérance de vie.

Deux autres méthodes, *Afficher* et *Diagnostic*, facilitent la compréhension de l'état de chaque plante. La première donne une vue d'ensemble de ses caractéristiques, tandis que la seconde propose un retour précis sur sa santé et ses besoins environnementaux, permettant ainsi au joueur de prendre des décisions éclairées. D'autres méthodes spécialisées, comme *RecupererEmoji* (pour l'affichage visuel avec des emojis qui était initialement prévu), *EstMature* (pour savoir si une plante porte des fruits), ou *InstancierMaladieAssociee*, renforcent l'interaction entre les plantes et leur environnement.

En résumé, la classe *Plante* agit comme une entité vivante autonome, qui réagit aux conditions de son environnement, vieillit, tombe malade, ou prospère selon les soins prodigués. Elle cristallise les principaux enjeux du jeu : observation, adaptation, et anticipation. Chaque tour devient ainsi une opportunité d'apprentissage, où l'on cherche à comprendre les besoins de ses plantes pour mieux les cultiver.

Saison

La classe abstraite *Saison* joue un rôle central dans la gestion des conditions climatiques du jeu. Elle représente le cadre environnemental dans lequel évoluent les plantes, en déterminant semaine après semaine les principaux facteurs météorologiques : pluie, ensoleillement et température.

Chaque saison est caractérisée par deux valeurs moyennes — un taux de pluie et un taux de soleil — qui servent de référence pour juger les conditions hebdomadaires. À chaque tour de jeu (représentant une semaine), la méthode *CalculerMeteo* est appelée pour générer aléatoirement un climat pour la période en question. Les valeurs de pluie et de lumière sont tirées au hasard dans l'intervalle [0,1], tandis que la température est choisie dans une fourchette définie par les sous-classes concrètes (Hiver, Printemps,

Été, Automne), à travers les méthodes abstraites *RecupererTempMin* et *RecupererTempMax*.

La classe propose également une méthode *DecrireMeteo*, qui donne une brève interprétation qualitative du temps (ex. « Nuageux », « Pluvieux », « Averse ensoleillée ») en croisant les niveaux de pluie et de lumière actuels avec leurs valeurs moyennes. Ce retour textuel permet au joueur de se faire rapidement une idée du climat de la semaine sans devoir interpréter directement les chiffres.

Enfin, la méthode *ToString* fournit une représentation textuelle complète des conditions climatiques en cours, utile à l'affichage dans l'interface console du jeu.

En résumé, *Saison* structure le cycle climatique de la simulation. En introduisant une part d'aléatoire contrôlé, elle permet de varier les conditions de culture tout en maintenant une cohérence propre à chaque saison. Les fluctuations météorologiques deviennent alors un élément clé de la stratégie de plantation, obligeant le joueur à s'adapter aux caprices de la nature.

Simulation

La classe *Simulation* est le centre de tout le jeu. C'est elle qui gère l'évolution de la partie semaine après semaine, en regroupant la grille de jeu, le calendrier des saisons, l'inventaire du joueur, et les maladies qui peuvent apparaître. Quand on crée une simulation, elle initialise ces différents éléments, en partant d'une grille vide et d'un inventaire de départ, avec un mode de jeu classique.

La partie commence avec la méthode *LancerJeu*, qui accueille le joueur et lui demande de choisir une saison pour démarrer. À chaque nouvelle semaine, la simulation vérifie si elle doit continuer en mode classique ou passer en mode urgence. Tous les cinq tours, un kart apparaît soudainement, détruit une ligne entière, et répand des maladies sur le terrain, ce qui ajoute un peu de tension à la partie.

En mode classique, *LancerTourClassique* déroule le jeu normalement. La météo est calculée selon la saison en cours, avec des valeurs aléatoires mais réalistes pour la pluie, la lumière et la température. Ces conditions influencent directement la croissance des plantes. Ensuite, la simulation met à jour chaque case : elle vérifie l'état des plantes, leur croissance, leur santé, et l'apparition éventuelle de maladies. Certaines maladies, comme la Toxinelle, peuvent ralentir ou même tuer les plantes si elles ne sont pas soignées.

Après cela, c'est au tour du joueur d'agir via le menu d'actions. Il peut arroser ses plantes, récolter des fruits, planter de nouvelles graines, ou utiliser des objets de son inventaire pour soigner ou booster ses cultures. Ces décisions influencent la suite de la partie, ce qui demande un minimum de stratégie.

Quand une urgence arrive, la méthode *LancerModeUrgence* prend le relais. Un kart fonce à travers la grille et peut détruire une ligne. Si le joueur a un éclair en réserve, il peut tenter de l'arrêter. Sinon, la ligne est détruite, et en plus des maladies sont propagées autour, ce qui complique la situation. Ce mode met un peu de pression et oblige à réagir rapidement.

La simulation permet aussi beaucoup d'interactions avec les plantes. Arroser réduit leur soif, récolter des fruits rapporte des objets, et utiliser l'inventaire est souvent nécessaire pour faire face aux maladies ou aux mauvaises conditions météo. Il est aussi possible d'analyser en détail l'état des plantes ou de consulter des informations sur chaque type de plante.

Enfin, la gestion des maladies est bien pensée. Les maladies apparaissent à certains moments, sont placées sur la grille, et se propagent dans les cases voisines. Cela oblige le joueur à rester attentif et à réagir pour éviter que la maladie ne se répande trop.

En résumé, la classe *Simulation* organise tout ce qui se passe dans le jeu. Elle gère le déroulement des semaines, les événements, les interactions avec les plantes, et les problèmes à gérer. C'est elle qui fait vivre le jeu et donne au joueur les moyens de prendre des décisions pour faire grandir son jardin.

Terrain

La classe *Terrain* sert de base pour représenter un type de sol ou d'environnement dans le jeu. Elle est assez simple et surtout pensée pour être étendue, car elle est abstraite. Chaque terrain a un nom, qui peut être défini pour identifier facilement le type de sol, comme une terre fertile, un sol sec ou un terrain marécageux.

La classe ne fait pas grand-chose toute seule, mais elle propose une méthode *ReagirASaison* que les classes qui en héritent peuvent personnaliser. Cette méthode permet de faire évoluer le terrain selon la saison qui passe, par exemple en modifiant ses caractéristiques ou en influençant la croissance des plantes dessus.

En résumé, *Terrain* est un modèle qui sert à créer des types de sols spécifiques, chacun pouvant réagir différemment aux changements de saison. C'est un point de départ simple mais flexible qui laisse la place à la créativité pour enrichir le jeu.

Relations entre les classes

Associations, héritages, compositions.

Le bon fonctionnement de notre simulation repose sur une organisation cohérente des classes, à travers plusieurs types de relations.
Parmi celles-ci, on distingue principalement l'héritage, les associations fonctionnelles ainsi que les compositions fortes. Chacune de ces relations joue un rôle structurant dans le système et reflète un certain niveau de dépendance ou d'interdépendance entre les objets.

Héritage

Plusieurs de nos classes sont abstraites, c'est-à-dire jamais instanciées. Elles servent de modèles communs à de nombreuses autres.

La classe *Plante*, par exemple est parente des classes :

- *ArbreACheddar* : qui, lorsque ses fruits sont récupérés, permet d'obtenir du cheddar, ainsi que des éclairs, sérums et étoiles, a pour terrain préféré *Fromage*.
- *FleurDeFeu* : qui permet d'obtenir des boules de feu, a pour terrain préféré *Volcan*.
- *FleurDeGlace* : qui permet d'obtenir des boules de glace, a pour terrain préféré *Ocean*.
- *PlanteCarnivore :* qui ne donne aucun objet mais peut manger des *Toxinelles* chez les *ArbreACheddar* contaminés voisins, et n'a pas de terrain préféré.

La classe *Maladie* est parente des classes :

- *Glaglaose* : qui infecte les *FleurDeFeu* et réduit leur satisfaction
- *Quecalorose (se prononce Qué Calorose)* : qui infecte les *FleurDeGlace* et réduit leur satisfaction
- *Toxinelles* : qui infectent les *ArbreACheddar* et les tuent au bout de trois semaines si le joueur n'intervient pas

La classe *Terrain* est parente des classes :

- *ArcEnCiel* : terrain plutôt neutre
- *Fromage* : terrain préféré des *ArbreACheddar*, est censé fondre en *Ete*
- *Ocean* : terrain préféré des *FleurDeGlace*, est censé geler en *Hiver*
- *Volcan* : terrain préféré des *FleurDeFeu*

La classe *Saison* est parente des classes :

- *Automne* : saison préférée des *ArbreACheddar*
- *Ete* : saison préférée des *FleurDeFeu*
- *Hiver* : saison préférée des *FleurDeGlace*
- *Printemps* : saison préférée des *PlanteCarnivore*

Associations : interagir sans être dépendant

Les associations désignent des relations plus souples entre objets, souvent traduites par des appels de méthode ou des attributs partagés. Elles illustrent la façon dont les objets interagissent entre eux, sans être fondamentalement liés dans leur cycle de vie.

La classe *Simulation*, cœur du système, en est l'exemple principal : elle interagit avec une large variété de composants, notamment :

- un *Calendrier*, utilisé pour suivre le temps et gérer les changements de saisons,
- une *Grille*, qui représente le jardin et regroupe toutes les cases,
- un *Inventaire*, pour la gestion des objets récoltés ou utilisés,
- une liste de *Maladie*, qui affectent ponctuellement les plantes,
- et des références constantes aux instances de *Saison*, *Plante*, ou *Terrain*.

De même, une *Plante* peut être liée à :

- un *Terrain*, sur lequel elle est semée (terrain préféré pour la croissance),
- une éventuelle *Maladie*, si elle est infectée,
- un *Inventaire*, à travers lequel elle peut produire des objets à récolter.

Une *Maladie*, de son côté, n'existe que pour interagir temporairement avec une *Plante*, en fonction de sa position sur la grille. Elle n'est pas liée de manière permanente à une plante, mais peut en infecter plusieurs au cours du temps.

Ces relations illustrent un niveau de dépendance fonctionnelle, sans lien de propriété fort. Les objets interagissent, mais peuvent exister indépendamment les uns des autres.

Compositions : structurer des entités complexes

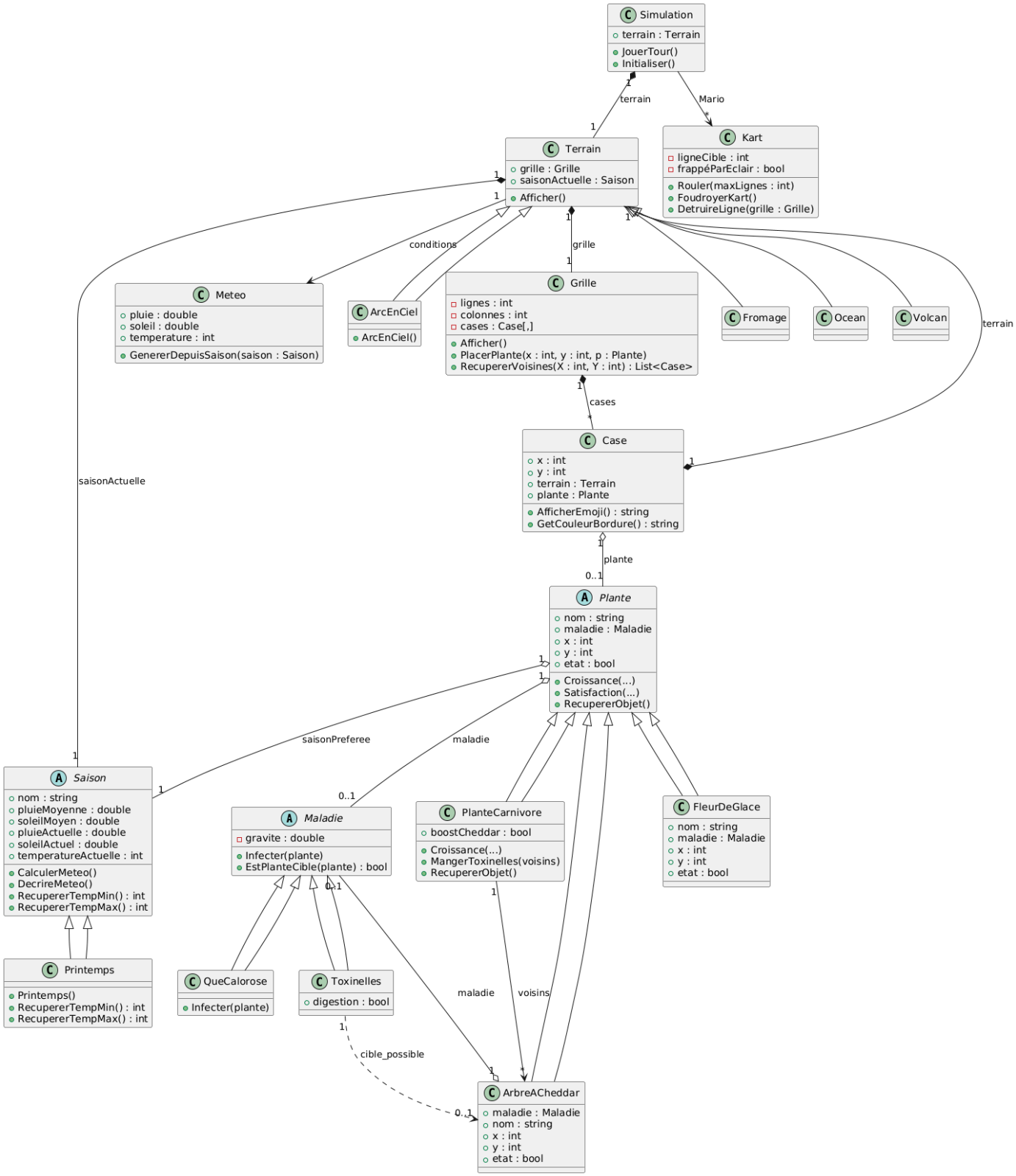
À l'inverse, certaines relations entre classes sont plus profondes : elles traduisent une composition, où un objet fait partie intégrante d'un autre. Dans ce cas, la durée de vie de l'un est intimement liée à celle de l'autre.

Par exemple :

- Une *Grille* contient un tableau de *Case* (souvent fixe et créé au départ). Chaque *Case* n'a de sens que dans le contexte d'une grille donnée.
- Une *Case* peut contenir une *Plante*. Si la case est supprimée ou modifiée, la plante disparaît avec.
- La *Simulation*, enfin, agrège des éléments comme une *Grille*, un *Calendrier*, un *Inventaire*, ainsi qu'une liste de maladies. Ces composants sont créés au lancement de la partie et vivent intégralement dans le contexte de cette simulation.

Cette logique de composition permet de regrouper les entités qui forment un tout cohérent. Le lien entre les classes est ici bien plus fort que dans une simple association : il traduit une appartenance, voire une dépendance structurelle.

Diagramme UML



IV. Tests de fonctionnalité

Nous avons fait le choix de citer seulement quelques fonctionnalités parmi toutes celles que notre jeu possède.

Fonctionnalité	Résultat	Remarques
Inventaire dynamique	OK	L'inventaire se met bien à jour quand on gagne un objet ou qu'on en utilise un
Croissance automatique des plantes	OK	Les plantes évoluent chaque semaine selon leur satisfaction
Récolte conditionnée	OK	On ne peut cueillir des fruits sur une plante uniquement quand celle-ci est mature
Affichage visuel de maturité	OK	Une plante est affichée en jaune quand elle est immature et en vert quand elle est mature
Usage sérum, étoile, cheddar	OK	Ces objets provoquent bien les effets escomptés
Usage boules de feu et de glace	A développer	/
Contagion à proximité	OK	Une maladie affecte les plantes dans un rayon de 1 case
Toxinelles	OK	Tue bien en 3 tours, si la maladie n'est pas soignée et peut être mangé par une plante carnivore
Kart	OK	Roule sur une ligne au hasard et tue les plantes qui s'y trouvent
Génération de maladies en mode Urgence	OK	A la fin du mode Urgence, quatre maladies apparaissent aléatoirement sur la grille
Changement automatique de saison	OK	Quand une saison se termine, la suivante débute
Météo aléatoire par saison	OK	Soleil, pluie et température variant selon la saison
Réaction des terrains par saison	A développer	/

V. Gestion de projet

Afin de travailler sur ce projet, nous avons fait deux réunions ensemble, afin d'imaginer comment implémenter les attendus de ce projet selon une logique de jeu qui nous serait propre. Cela nous a permis très tôt de nous séparer les différentes classes et sous-classes associées, permettant de travailler sur GitHub à distance. Par ailleurs, nous avons fait usage du README qui a été créé à ce moment-là pour nous aiguiller tout au long du projet, et ne pas oublier les différentes fonctionnalités dont nous avions discuté. Le projet se déroulant sur un temps assez long, celui-ci nous a été d'une grande aide afin de ne pas perdre de temps pour se remettre à travailler après une période où d'autres projets ont été préférés.
Les différents créneaux en classe nous ont donc permis de travailler sur la logique générale du jeu et la manière dont celui-ci allait se dérouler (ordre d'appel des différentes fonctions dans Simulation.cs notamment).

VI. Bilan

Nous sommes dans l'ensemble satisfaits de notre projet. Le jeu que nous avons développé est ludique et reste agréable à jouer. Cependant, il présente plusieurs lacunes importantes.

En ce qui concerne les fonctionnalités, un certain nombre de celles que nous avions initialement envisagées n'ont pas pu être mises en œuvre. Cela s'explique en grande partie par un excès d'ambition de notre part, ainsi qu'une approche parfois trop précipitée dans la conception de nombreuses classes, sans effectuer de tests réguliers. Ce manque de rigueur a conduit à la création de nombreuses classes qui, bien qu'elles existent, nécessiteraient un développement plus poussé pour réellement enrichir le jeu, ce qui est le cas par exemple pour les différents types de terrain.

Avec le recul, nous reconnaissons qu'une approche plus mesurée, en implémentant les fonctionnalités progressivement et en procédant à des tests plus fréquents, aurait été plus judicieuse. Cela nous aurait permis d'éviter la surcharge de travail et d'assurer une meilleure cohérence dans le développement du projet.