Programmation avancée - Projet 2025

ENSemenC

Image générée par Copilot

Contexte

Le jeu est un simulateur de potager. Il se déroule dans un pays choisi avant le lancement (qui peut être imaginaire), ce qui conditionne le type de semis accessibles et le type de météo. Le joueur dispose initialement d'un certain nombre de semis et d'un certain nombre de terrains possédant des qualités fixées.
Le jardin est observé en permanence via une webcam qui envoie des alertes au joueur en cas de fortes intempéries ou de la présence d'un intrus.

Fonctionnalités minimales attendues

- Une classe abstraite rassemblera les caractéristiques communes aux **Plantes**. Une autre classe abstraite rassemblera les caractéristiques communes aux **Terrains**. Les classes dérivées de ces classes de base devront être associées judicieusement.

- Les semis/plants seront caractérisés par leur nature (annuelle ou vivace, comestibles ou non, mauvaises herbes…), leur(s) saison(s) de semis, leur terrain préféré (sable, terre, argile), l'espacement entre elles et la place dont elles ont besoin pour grandir (attention aux espèces envahissantes), leur vitesse de croissance, leurs besoins en eau et en luminosité, leurs zones de températures préférées, les maladies qu'elles peuvent attraper (avec une probabilité de contamination tirée aléatoirement), leur espérance de vie (si rien de fâcheux ne leur arrive avant), le nombre de pousses/fruits/légumes que peut donner un plant.
- Si les conditions préférées de la plante ne sont pas respectées à au moins 50% elle meurt. Plus les conditions préférées sont réunies, plus la plante pousse vite et donne de produits.
- Vous avez aussi la possibilité de cultiver des plantes non comestibles mais commercialisables (coton, bambou…) ou des plantes ornementales (fleurs, cactus, plantes carnivores, plantes imaginaires ou de conte de fées…)
- Le jeu comportera une **double temporalité**:
    - Il simulera le passage du temps à partir d'une date donnée, pendant lequel les plantes semées par le joueur poussent et sont observées de semaine en semaine ou de mois en mois = **Mode classique**
    - Il donnera une image dynamique simplifiée de l'état du jardin en temps réel en cas d'intempérie ou de la présence d'un intrus = **Mode urgence**
- Le mode classique sera donc le mode "de croisière" avec l'affichage des plantes qui poussent à leur rythme, le mode d'urgence interviendra de façon aléatoire à l'écran en signalant au joueur une urgence à traiter et affichera alors le potager en temps réel avec l'intempérie ou l'intrus en train de se déplacer.
- **Mode classique:**
    - Les tours de boucle du jeu dans ce mode simulent des laps de temps relativement longs : des semaines a minima.
    - À chaque tour, le joueur peut effectuer des actions pour s'assurer de la bonne santé de son potager (désherber, pailler, arroser, traiter, semer telle ou telle graine, récolter un légume mûr, installer serre, une barrière, un pare-soleil…)
    - Le jeu déploie une simulation semi-réaliste mais simplifiée de la météo au fil des laps incluant les taux de précipitations (qui assurent que les plantes ont assez d'eau, mais aussi potentiellement trop !) et les variations de température (attention si les plantes gèlent ou subissent une sécheresse).
    - Le jeu introduit des "obstacles" et des "bonnes fées": des maladies, des pucerons, des larves de coccinelles (qui mangent les pucerons), des escargots, des rongeurs, des oiseaux, des lagomorphes, des talpinae, des piétineurs, etc, qui peuvent impacter la santé des plantes.
    - L'affichage à chaque tour résume les plantes en place, leur santé, leur hauteur, de préférence avec des dessins en console qui ressemblent aux plantes dans différents états, les montrant en place sur leur terrain, et offre un menu d'actions à effectuer. Le "tour-semaine" est fini quand le joueur le décide, ou quand il a accompli son nombre limité d'actions

- **Mode urgence:**
    - Les tours de boucle du jeu en mode temps réel peuvent simuler les déplacements d'un rongeur en train de manger les récoltes ou des intempéries intenses telles qu'une tombée de grêle pour pouvoir mettre en place des solutions d'urgence.
    - À chaque tour, le joueur peut effectuer des actions d'urgence proposées dans un menu dédié, pour assurer la survie de son potager (faire du bruit, déployer une bâche, fermer une serre, acheter un épouvantail, reboucher des trous, creuser une tranchée…) Attention: interdit de tuer les animaux dans ce jeu!

Fonctionnalités bonus

- **Mode Magasin:** Le jardinier pourra ensuite faire croître sa plantation en vendant ses récoltes ou en transformant une partie des graines/fruits/ boutures récoltées en nouveaux semis pour l'année suivante. On ajoute un système d'"argent" ou de valeurs échangeables (réel ou imaginaire) qu'on peut dépenser et qu'on peut gagner. Idées d'utilisation de l'"argent": achat de parcelles de terre pour agrandir la plantation, achat de matériel (pelles…), achat de choses pour protéger les récoltes (serres, barrières, voilages…). On fera un bilan comptable ou le calcul des profits/pertes/rentabilité quand le joueur demande à quitter le jeu.
- **Mode IA**: Au moment de l'affichage: proposer des recommandations selon les actions qu'il serait pertinentes de faire à un moment donné
- **Mode Ecologique**: Agriculture biologique, bilan carbone, calcul de l'impact environnemental, rotation des récoltes, désherbage à l'aide d'animaux…
- **Sauvegarde** de la partie en cours
- **Autres**… Soyez créatifs !

Exigences techniques

Voici les exigences techniques associées au projet :

- Il prend la forme d'une application Console en C#, obligatoirement créée avec la version 8 du framework .NET.
- Le code met en œuvre à bon escient les mécanismes de la programmation orientée objet : encapsulation, associations entre classes, héritage, classes abstraites, polymorphisme, etc.
- Les listes sont privilégiées aux tableaux pour gérer les collections d'objets.
- La duplication de code est limitée au maximum.
- Le code est correctement indenté et commenté aux endroits nécessaires.
- Le code respecte la convention de codage C#.

Organisation

Le projet est réalisé en binôme avec GitHub comme plateforme obligatoire de partage du code. Les binômes intra-groupes sont à privilégier. Cliquez sur le lien suivant pour créer ou rejoindre une équipe: (vous devez disposer d'un compte GitHub).

https://classroom.github.com/a/V4LfdTHJ

Un dépôt Git sera automatiquement créé pour votre projet dans l'organisation GitHub PROGAV-PRJ25. Ensuite, clonez ce dépôt sur votre machine locale pour commencer à travailler. **Le nom du dépôt doit comprendre les noms de tous les membres du binôme afin de pouvoir les identifier facilement**.

Planning et livrables

La date limite pour ce projet est fixée au vendredi 23/05/2025 à 23h59. (Attention, les dépôts de l'organisation PROGAV-PRJ25 sont configurés de façon à ce qu'aucune modification ne soit possible au-delà de la deadline).

Le **code source** sera publié sur GitHub, dans le dépôt du projet.

Afin de préserver la santé mentale des correcteurs, il est **TRÈS IMPORTANT** que le projet soit aussi facile à tester que possible. Le code déposé sur GitHub doit permettre, après clonage, de lancer l'application sur un nouveau poste sans aucun problème. **Merci de faire cette vérification après votre dernier *commit*.** Tout projet présentant des difficultés d'installation sera lourdement pénalisé.

Vous devez par ailleurs rendre un code qui compile sans aucun warning.

Vous devez également rédiger un **rapport technique** décrivant :

- l'univers du jeu et ses spécificités ;
- les possibilités des joueurs et le déroulement d'une partie ;
- la modélisation objet réalisée (diagramme de classes UML bienvenue) ;
- les tests réalisés pour vérifier le bon fonctionnement de l'application ;
- la gestion de projet (organisation de l'équipe et planning de réalisation) ;
- un bilan critique sur le projet.

Ce document sera ajouté au format PDF à la racine du dépôt GitHub du projet.

En complément, la **matrice d'implication** située à la racine du dépôt devra également être complétée.

Critères d'évaluation

- Richesse fonctionnelle, y compris plus-values et originalité.
- Fluidité et lisibilité de la simulation.
- Qualité technique de la réalisation :

    - Qualité de la conception : utilisation des mécanismes objets à bon escient, absence de duplication de code, etc.
    - Qualité du code source : indentation, commentaires, respect de la convention de codage C#, etc.
    - Absence de bug ou plantage.
- Qualité du rapport (fond et forme).
- Respect des délais imposés.

ATTENTION : votre projet devra obligatoirement être une création originale implémentée par vos soins, tout manquement sera lourdement sanctionné.

Image générée par Copilot