Table des matières

Informations générales

- Nom : Autonomie.
- Durée : semaines 1 à 6 soit exactement 24 jours.
- Résumé : Développer un jeu (console) mettant en scène l'avatar arrivant sur un nouveau territoire (l'île, dont il n'a accès qu'à une petite portion), il doit y découvrir de nouveaux lieux, une faune, une flore, des ressources et des énigmes à résoudre.
- Objectifs : l'avatar doit pouvoir gagner le jeu afin de passer au niveau (projet) suivant

Compétences mises en œuvre

- Personnelles :
  - Exploration
  - Découverte
  - Adaptation
  - Autonomie

- Organisationnelles et fonctionnelles :
  - Compréhension d'un contexte
  - Organisation d'un projet
  - Communication (possibilité de communication illimitée avec les autres membres du groupe)
  - Travail en binôme (début de collaboration en équipe)

- Techniques :
  - Algorithmique
  - Développement séquentiel et procédural
  - Python (tous les principes et syntaxes de base, incluant les variables, les types, les test, les boucles, les opérateurs- mathématiques, de comparaison et logiques-, les interactions de base - print, input -, les fonctions - avec et sans paramètres, return -, les listes (y compris à 2 dimensions)/tuples/dictionnaires, la sérialisation, les fonctions de string, la lecture et l'écriture de fichiers texte et JSON, le parsing de fichier, les modules- os, random, json, …-, des mots clés - in, continue, break, None, …-, la gestion des exceptions try/except, …)
  - Environnement de développement, IDE (incluant linting et debugging)
  - Conceptualisation (diagramme de flux, product backlog de niveau 1)

Spécifications fonctionnelles

Le jeu doit permettre de :

- Demander le nom du joueur.
- Représenter en mode texte (console) la carte (résolution minimum 50x30, la carte sera statique, c'est-à-dire qu'elle tient entièrement sur l'écran sans défilement) qui t'as a été remise avec tous les éléments indiqués (eau, plaine, arbres, rochers, endroits mystérieux- 4 en tout, etc.).
- Représenter ton avatar sur cette carte.
- À l'avatar de se déplacer sur la carte selon 4 directions en respectant les obstacles.
- Gérer le contenu de son sac à dos (avec une capacité d'objets maximum).
- Gérer une liste d'objets à trouver sur la carte (avec possibilité de les placer aléatoirement en début de partie), de pouvoir les ramasser (et les mettre dans le sac à dos) ou les redéposer au sol.
- Gérer l'utilité de ces objets, certains seront utiles pour certaines actions spécifiques, d'autres juste pour prendre de la place dans le sac à dos (ou peut-être à utiliser pour les étapes/projets suivants).
- Gérer 3 niveaux vitaux (soif, faim et fatigue) de l'avatar sur une échelle de 0 à 100. Si l'un des compteurs arrive à zéro, c'est cuit, la partie est perdue !

Spécifications additionnelles

- À chaque déplacement sur la carte, l'hydratation et la satiété diminuent de 2 et l'énergie (inverse de la fatigue) de 3.
- L'avatar peut être mis en sommeil pour un nombre d'heures déterminé, pour chaque heure passée à dormir, l'hydratation diminue toujours de 2 (il fait vraiment chaud sur l'île), la satiété de 1 (c'est bien connu, qui dort dine) et l'énergie remonte de 6.
- On doit gérer 1 compteur pour le nombre de déplacements et 1 pour le nombre d'actions (autres que déplacement) effectués depuis le démarrage de la partie.
- De la nourriture pourra être trouvée sur la carte, apportant un pourcentage de satiété lorsqu'elle est consommée (attention à la nourriture non comestible 😉).
- De l'eau pourra être trouvée sur la carte, apportant un pourcentage d'hydratation lorsqu'elle est bue (attention à l'eau salée), la bouteille en verre pourra également être remplie (ou vidée ou bue).
- Lorsque l'avatar arrive à un emplacement mystérieux, cela correspond à un défi à relever (un petit jeu à programmer dans le jeu), la réussite du défi donne un objet particulier à l'avatar (les notions essentielles à la réalisation de ces défis sont traitées lors des dojo et live coding).
    - Défi 1 : Le nombre mystérieux. Donne la clé de bronze.
    - Défi 2 : Le code César. Donne la clé d'argent.
    - Défi 3 : Le multi Fizz-Buzz. Donne la clé d'or.
    - Sortie : le quatrième lieu contient la porte d'accès au projet 2 et nécessite les 3 clés pour être ouverte.
- Lorsque le jeu est arrêté, il doit sauvegarder l'état exact de la carte (objets présents, lieux explorés ou non, énigmes résolues ou non), l'emplacement de l'avatar, ses niveaux vitaux, les compteurs de déplacement et d'actions et le contenu de son sac à dos.
- Lorsque le jeu est (re)lancé, on doit pouvoir choisir de continuer la partie en cours (1 seule sauvegarde) ou de démarrer une nouvelle partie avec les paramètres initiaux (carte initiale, position et niveaux vitaux de l'avatar, contenu du sac à dos).
- Lorsque la partie est gagnée (ou perdue), on sauve le score (date et heure, nom du joueur, gagné/perdu, les compteurs, les niveaux vitaux).
- En début de partie, on propose d'afficher l'historique des parties, trié par date/heure décroissante
- Enfin, le jeu doit être "modable", donc aucune information ne doit être codée "en dur" dans l'application. En clair cela signifie que, sans toucher une seule ligne de code, on doit pouvoir lui fournir :
    - une nouvelle carte
    - de nouveaux objets à de nouveau emplacements et avec leur éventuel impact sur les signes vitaux
    - une nouvelle échelle de signes vitaux et d'évolution de ces signes lors d'un déplacement, d'une autre action, d'une heure de sommeil
- Évidemment, il ne sera pas possible de gérer des comportements non prévus (par exemple "miner" une ressource, construire un bâtiment, combattre, …)

Spécifications techniques

- L'application est documentée. C'est-à-dire que, outre les spécifications ci-présentes, doivent être inclus dans le dossier Documents :
    - un diagramme de flux- avec sous-diagrammes- clair et bien légendé
    - une liste des tâches à effectuer (product backlog) avec priorisation et dépendances (mais pas forcément rédigées en User Stories)
- Le code respecte autant que faire se peut la PEP8 et les bonnes pratiques (DRY, …)
- Les fichiers de l'application sont correctement organisés (dossiers spécifiques- Documents, Resources, ProgramFiles, …)
- Tous les nommages (dossiers, fichiers, variables, constantes, fonctions, etc.) doivent être en anglais
- Le code doit être commenté (docstrings et #) et aéré (lignes blanches)
- L'application doit utiliser des fonctions (def) et être développé en mode procédural
- Le programme principal ne doit contenir que des appels à des fonctions (pas de code direct)
- Il faut autant que possible éviter l'utilisation du mot clé "global"
- L'application doit offrir la meilleur expérience (UX/UI) possible à l'utilisateur (chaque action effectuée doit afficher un message explicite, elle doit être visuellement agréable- autant que faire se peut en console, intuitive, addictive- disons un minimum pour vouloir arriver à la fin au moins une fois :-D)
- Le package de l'application (incluant la documentation) doit être considéré par les autres développeurs au minimum comme acceptable pour être reprise (pour maintenance et évolutions)
- L'application est développée en environnement virtuel
- Le package est publié sur GitHub sous licence libre

Améliorations possibles

Si tu es en avance sur le projet, tu peux ajouter les spécifications suivantes :

- Fonctionnelles :
    - La représentation de la carte peut tirer parti des caractères semi-graphiques de la table ASCII étendue en place des seules lettres, chiffres et signes standards (attention, le fichier contenant le plan de la carte doit lui n'être constitué que de signes standards)
    - Toutes les infos représentées peuvent utiliser du texte riche (couleurs, positionnement absolu) grâce à l'implémentation de codes "escape"
    - La carte peut être représentée dans une résolution dépassant la taille de l'écran (elle devra donc pouvoir défiler lorsque le personnage se déplace, ce dernier sera centré tant que la carte peut défiler, ou s'approcher des bords de l'écran lorsque la bordure de la carte coïncide avec la bordure de l'écran)
    - Les lieux mystérieux peuvent faire l'objet de sous-cartes dans lesquelles il est également possible de se déplacer (par exemple une grotte), donc il convient de gérer les changements de carte (entrée/sortie)
    - Gérer plusieurs sauvegardes, associées à des joueurs distincts
    - Inventer une formule mathématique permettant de calculer un véritable score en fin de partie (et pas seulement l'affichage des différents paramètres)
    - Ajouter de nouveaux lieux mystérieux et de nouveaux défis

- Techniques :
    - Le code peut être développé en respectant les principes de Programmation Orientée Objet (normalement vu et appliqué dans le cadre du projet 2)
    - Les tâches du Product Backlog peuvent être rédigées comme des User stories (normalement vu et appliqué dans le cadre du projet 2)

Conditions de réussite

- Le jeu est considéré terminé :
    - s'il répond à toutes les spécifications fonctionnelles et techniques
    - et s'il est gagnable en démarrant une nouvelle partie (avec les paramètres initiaux)

- Le projet est validé si, en plus du point précédent, tu es en mesure d'expliquer :
    - ta démarche
    - tes documents
    - ton code
    - tes choix d'architecture…

bref si tu maitrises parfaitement ta solution.