Projet

Labyrinthe

Par équipe de 3, vous êtes chargé de la réalisation d'un jeu vidéo et de sa promotion au travers d'un site web.
La structure du projet vous est imposée :

- Vous utiliserez un moteur de jeu en python déjà conçu basé sur **Pygames**.
- Votre site web sera basé sur Flask.
- Vous documenterez votre code avec pydoc.
- Vous réaliserez des tests unitaires avec pytest.
- Vous utiliserez une méthodologie TDD

Le dossier complet est accessible à l'adresse https://github.com/aulery/NSI_labyrinthe

L'arborescence du dossier est la suivante :

```
❖ NSI_Labyrinthe : Contient les documents globaux du projet.
   ➢ Code_jeu : Contient le code python du jeu
      ▪ images : Contient les images qui seront utilisée pour le jeu.
      ▪ jeu_de_test : Contient les codes python des test unitaires fait avec Pytest
   ➢ Site_Web : Contient un exemple minimal d'un site web gérer avec Flask
      ▪ static : Contient les images du site web
            ➢ Documentation : contient la documentation autogénérée par python
         ♦ templates : Contient les modèles utilisés par Flask
```

I. Déroulement du projet :

Semaine 1
- Découverte du projet, constitution des groupes (1 heure).
- Lecture du cahier des charges, test de la maquette fournie (1 heure).
- Détermination des nouvelles fonctionnalité, amélioration du cahier des charges, répartition des taches

Semaine 2
- Traduction du cahier des charges en algorithmes (2 heures)
- Rédaction des tests de validations et vérification papier des algorithmes (2 heures)

Semaine 3
- Implémentation des tests en python (1 heure)
- Codage des nouvelles fonctionnalités (2 heures)
- Validation et documentation (1 heure)

Semaine 4
- Design des 3 pages du site web. (1 heure)
- Codage des modèles de pages du site (1 heure)
- Mise en place des fonctionalités Flask (2 heures)

Semaine 5
- Test des fonctionnalités de la maquette et nettoyage complet du code (2 heures).
- Rédaction du manuel du projet (2 heures).
- Rendu projet avec manuel, algorithmes, tests, design web et cahier des charges modifié.

II. Cahier des charges :

Le logiciel de jeu doit incorporer les nouvelles fonctionnalités suivantes.

	Fonctionnalité	Critère	Niveau	Flexibilité
1	Sauvegarder un niveau dans un fichier	Format de fichier Contenu	Csv Complet	Ou XML/JSON Statistiques additionnelles
2	Charger un niveau depuis un fichier	Format de fichier Contenu	Csv Complet	Ou XML/JSON Statistiques additionnelles
3	Déplacer le personnage	Déplacement sur l'appui des flèches directionnelles	Une touche à la fois	Aucune
4	Permettre une action spéciale	Déclenchée à l'appui de la touche espace.	Une action	Conditionnelle ou non.
5	Permettre au joueur de choisir son nom	Longueur Encodage	16 Caractères UTF8	10 minimum Aucun
6	Permettre une interaction avec un objet	Action au passage du personnage	Au moins une	Immédiate ou permanente
7	Exporter les données joueurs dans un fichier.	Format de fichier Contenu	Csv Nom score date statistiques	Ou XML/JSON Avec ou Sans statistiques
8	Calculer un score du joueur	Score disponible en fin de partie	Chiffre ou lettre	Aucune
9	Prendre en compte les murs	Possibilité de déplacement	Impossible	Par action spéciale
10	Offrir le Choix de charger/sauvegarder un niveau	Possibilité offertes	Consoles	La sauvegarde peut être à part.

Le site web doit avoir les caractéristiques suivantes :

	Fonctionnalité	Critère	Niveau	Flexibilité
1	Avoir une page de présentation	Contenu Mise en forme	Présentation du logiciel + Menu Sur 3 Niveau	Aucune
2	Avoir de page de résumé des scores	Contenu Mise en forme	Nom score date statistiques Sur 3 Niveau	Aucune
3	Avoir une page d'index de la documentation.	Contenu Mise en forme	Page auto-générée Sur 3 Niveau	Génération manuelle à partir d'un Template
4	Avoir accès à la page des scores depuis la page de présentation	Accès par requête	POST ou GET	Aucune
5	Avoir accès à la documentation depuis la page de présentation	Accès par lien	Hypertexte	Formulaire ou autre
6	Permettre de trier la page des scores	Méthode de tri	3 critères	3 minimum
7	Importer les données joueurs depuis un fichier.	Format	CSV	Ou XML/JSON Avec ou Sans statistiques

III. Le moteur de jeu :

Le logiciel est composé de plusieurs fichiers, en rouge sont indiqués les fichiers qui sont censés être finalisés. Vous devez lire attentivement les fichiers Actions.py, Regles.py et main.py

- main.py : Fichier principal, c'est là que l'on indique quels sont les règles, images, carte et joueur.
- Actions.py : Fichier qui permet de paramétrer les interactions avec les joueurs vous devrez compléter/modifier les 5 fonctions qui le compose.
- Regles.py : Fichier qui permet de paramétrer la réaction du jeu à la position du joueur sur la carte. Vous devrez compléter/modifier/ajouter les fonctions qui le compose.
- Carte.py : Fichier qui gère les cartes
- Etat.py : Fichier qui gère l'état de la partie.
- IHM.py : Fichier qui fait l'affichage graphique
- Joueur.py : Fichier qui gère un joueur
- Statistiques.py : Fichier qui permet le suivit des statistiques

Le moteur de jeu suit la procédure suivante :

1. Choix des éléments à inclure dans la partie (règles, images, niveau, personnage)
2. Affichage des éléments à leur position définit.
3. Prise en compte des Actions.
4. Applique les règles et si besoin reviens en arrière.
5. Retour à 2. (Sauf si la règle victoire indique que la partie est finie)
6. Traitement en fin de partie (sauvegarde, score, etc.).
7. Fin de programme

Pour ajouter une image, vous devez mettre votre fichier dans le répertoire dédié, celle-ci doit être de dimension **20x20** pixels.

IV. La documentation :

La documentation de votre code de jeu, sera générée automatiquement **à partir de vos docstring** et placée dans le répertoire /site_web/static/documentation.

Pour faire ceci, placer vous dans le répertoire de base du projet et exécuter le script

```
generer_documentation_(linux/win).sh
```

V. Les tests unitaires :

Les tests unitaires sont à réaliser avec la bibliothèque pytest.

Celle-ci fonctionne très simplement, il faut se placer dans le répertoire Code_jeu et lancer la commande :

```
Python3 -m pytest -v
```

Celui-ci va rechercher et exécuter tous les fichiers commençant par test_ dans ces fichiers il exécutera toutes les fonctions commençant par test_.

Pytest affichera un rapport dans la console en indiquant clairement quel test a échoué et pourquoi.

Un jeu de test existe déjà pour les fonctions créées par l'enseignant, vous pouvez vous inspirez d'elles.