

Projet pyChase

Modalité du projet

Durée

- 5 semaines (jusqu'à la semaine 14 incluse)

Évaluation :

L'évaluation sera décomposée en deux parties : qualité et algorithmes.

Voici les sous-parties et le poids de chacune :

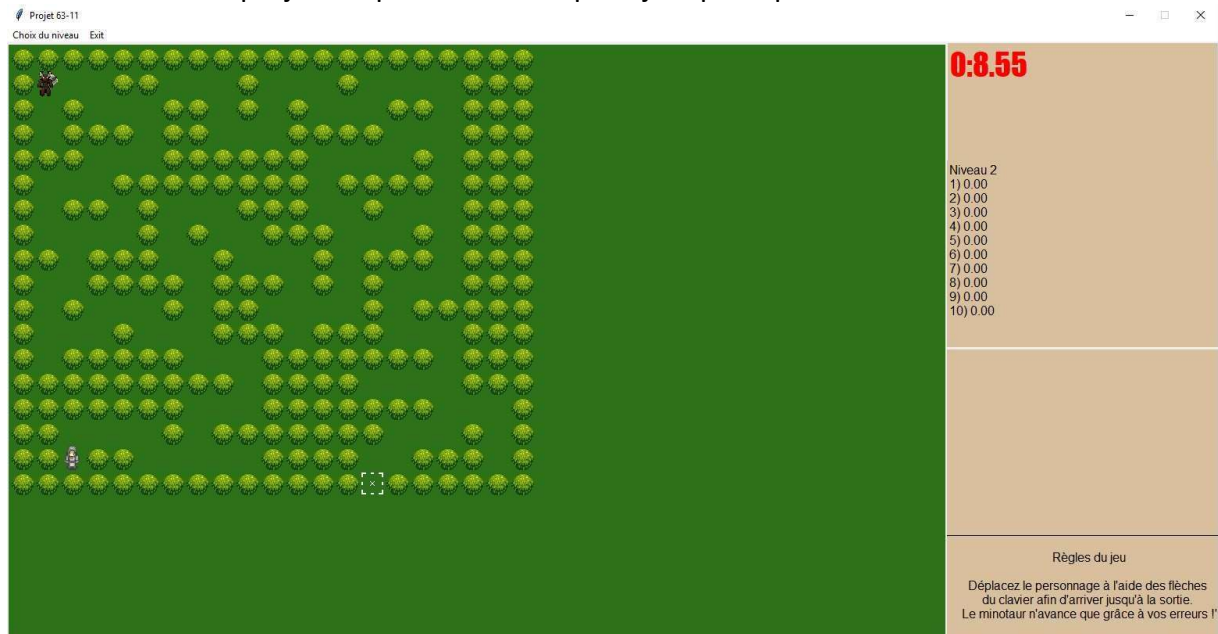
- Qualité du code (15%)
 - Docstring (6%)
 - Typage (variables, paramètres et types de retour) (5%)
 - Documentation du code complexe (4%)
- Fonctionnalité (35%)
- Logique et algorithmique (25%)
- Qualité du programme (25%)
 - Décomposition de fonctions et modules
 - Structures de données
 - Programmation défensive (test de paramètres, valeur nulle, etc.)

Temps à disposition




- Heures de laboratoire

Description du projet

L'objectif de ce projet est de mettre à profit toutes les notions apprises lors des cours du module 63-11. Le projet en question est un petit jeu qui se présente sous la forme suivante :



L'objectif du jeu

Le but du jeu est d'atteindre la sortie  dans le temps imparti sans se faire attraper par le minotaure . Le minotaure n'avance que lorsque le joueur commet une erreur. Par erreur nous entendons le fait que le joueur avance contre un mur . Par défaut, le minotaure avancera de 5 cases sur le chemin le plus court pour atteindre le joueur.

Quelques règles

- Lorsque le joueur atteint une sortie, il disparaît de l'écran et le jeu s'arrête.
- Lorsque le minotaure atteint un joueur, le joueur disparaît, le minotaure prend la position du joueur et le jeu s'arrête.

Vous devez prendre en compte les « Quelques règles » listées ci-dessus lors du développement des mouvements.

Quelques conseils :

- Essayez d'imaginer ce qu'implique un déplacement par rapport à une entité ainsi que sur vos structures de données.
- Essayez, à priori, de déterminer les 4 cas de déplacements possible (je ne parle pas de haut/bas/gauche/droit).

Semaine 13 : Scores

Finalement, dans le répertoire scores, fichier scores.txt, se trouvent les scores pour chaque niveau. Ceux-ci sont stockés sous cette forme :

<numNiveau>;score1;score2;score3;score4;score5;score6;score7;score8;score9;score10;

Exemple pour le niveau 1 : 1;10.24;11.34;11.85;11.86;12.02;13.57;14.99;0.0;0.0;0.0;

4 fonctions seront à développer pour la gestion des scores :

1. `chargement_score`
Fonction chargeant les scores depuis un fichier.txt et les stockent dans un dictionnaire.
2. `maj_score`
Fonction mettant à jour l'affichage des scores en stockant dans un str. C'est l'affichage visible sur la droite du jeu.
(*"Niveau x*
1) 11.25
2) ... ").
(*Niveau x*
1) 11.25
2) ... ").
3. `sauvegarde_score`
Fonction enregistrant un nouveau score réalisé par le joueur. Le calcul de score est le suivant :
$$\text{temps_initial} - \text{temps_restant}$$

Ce score est arrondi à 2 chiffres après la virgule et stocké en tant que float.
4. `update_score_file`
Fonction qui va écrire les scores dans le fichier scores.txt. Cette fonction est appelée lors de la fermeture de l'application

Fonctions et outils fournis

Dans le script `outils.py` vous trouverez toutes les fonctions à votre disposition. Voici la liste des fonctions que vous trouverez :

- `creer_image()`
- `creer_mur()`
- `creer_minotaure()`
- `creer_sortie()`
- `creer_personnage()`
- `creer_case_vide()`
- `coordonnee_x()`
- `coordonnee_y()`
- `est_egal_a()`

Pour comprendre ce qu'elles font, référez-vous à leur docstring. Pour les utiliser, vous pouvez directement les appeler dans votre code (en faisant attention de bien utiliser leurs paramètres dans le bon ordre).

Dans le fichier `fourni\personnage.py` vous trouverez des fonctions que l'on pourra utiliser sur l'entité Joueur.

- `est_mort()`
- `est_attraper()`
- `a_fini()`
- `est_sorti()`

Pour comprendre ce qu'elles font, référez-vous à leur docstring. Pour les utiliser, vous pouvez directement les appeler dans votre code sur votre entité joueur. Par exemple `joueur.est_sorti()`.

`liste_image : []` Vous l'utiliserez uniquement lorsque vous avez besoin de la fonction `creer_image()`. Chacun de ses index fait référence à une image :

- [0] : image du mur
- [1] : image de la sortie
- [2] : image du minotaure
- [3] : image de la caisse sur une cible
- [4] : image du joueur
- [5] : image du joueur sur la cible
- [6] : image du sol

Donc si je souhaite remplacer l'image par l'image du joueur -> `liste_image[4]`

`dict_score` qui est le dictionnaire stockant les scores. Celui-ci est structuré de cette manière : `key = numéro du niveau`, `value = tableau de scores ordonnés` (mais les 0 sont à la fin)

ainsi : `{ 1 : [10.24,11.34,11.85,11.86,12.02,13.57,14.99,0.0,0.0,0.0]}`