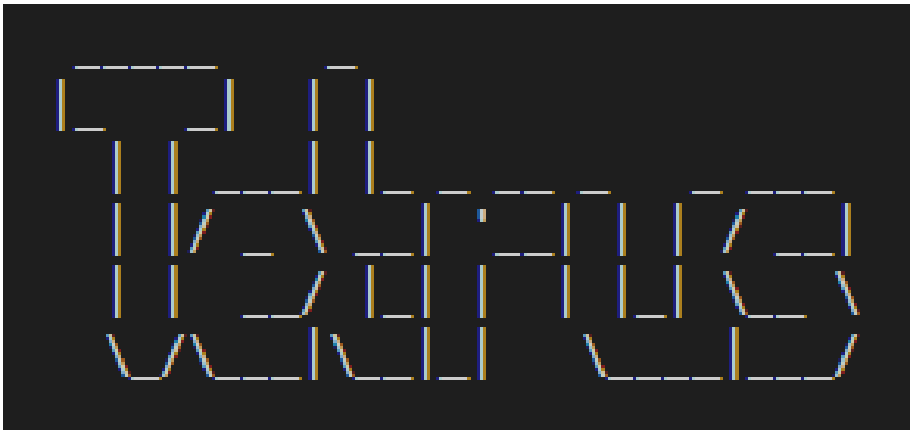
Rapport Projet Python Semestre 1 :



Guillaume DELHAYE — Idir NAIT MEDDOUR — Tomas TARGE
2022-2023

PARIS PANTHÉON-ASSAS UNIVERSITÉ

Table des matières :

Les objectifs

Le but principal du projet est de créer un jeu ressemblant au Tetris et se voulant polyvalent au niveau de ses règles. Pour se faire, l'utilisation de fonctions est nécessaire, on peut donc l'inclure dans les objectifs.

Les principaux paramètres du jeu sont :

- Le plateau, sa forme et sa taille
- Les blocs, leurs formes et leur mise à disposition

Ces deux paramètres sont polyvalents et déterminés par le joueur, mais plus encore, ils interagissent l'un l'autre lors du jeu. Lorsque l'on pose un certain bloc choisi selon une certaine règle, on doit vérifier qu'il ne sort pas du plateau.

Tout l'objectif du projet repose sur l'interaction d'éléments modulables et incite donc à leur optimisation afin de faciliter leurs échanges. Cela passe bien évidemment par la mise en place de fonctions.

Les fonctionnalités

Une grande partie des fonctionnalités sont visibles dans l'écran de paramétrage du jeu :

- Choix des formes du plateau (triangle, losange, cercle).
- Choix de la largeur du plateau (entre 21 et 26), agit sur la forme pour qu'elle remplisse le maximum d'espace donné.
- Choix du mode de sélection de blocs (trois blocs aléatoires parmi ceux autorisés ou bien tous les blocs autorisés).

L'autre partie est constituée des mécaniques primaires du jeu :

- Tombée des blocs quand remplissage d'une colonne.
- Plus de points ajoutés au score quand une grande ligne/colonne disparaît.
- Placement des blocs par coordonnées données par le joueur.
- Vérification de la validité de l'emplacement.

Les différents algorithmes

- `makeCircle(self):`

Génère une grille (une matrice 2D) de jeu en forme de cercle vide, de la taille définie par l'utilisateur avant le début de la partie.

-`makeLosange(self):`

Génère une grille (une matrice 2D) de jeu en forme de losange vide, de la taille définie par l'utilisateur avant le début de la partie.

-`makeTriangle(self):`

Crée un losange `makeLosange(self)` puis en coupe la moitié pour former un triangle.

-`fallBlocks(self):`

Vérifie l'ensemble des cases contenant un bloc de `self.grid` et si la case d'en dessous est libre fait tomber le bloc aussi bas que possible.

-`valid_position(self, bloc, i, j):`

- Vérifie que le bloc ne sort pas du plateau en horizontale ou en vertical.

- Vérifie si chaque case pleine de la matrice du bloc à poser est posable.

- Retourne un booléen issue de son analyse.

-`inputConfiguration():`

Interprète le choix de configuration de l'utilisateur, en accord avec `printConfiguration()`

-`print_grid(self):`

Affiche sur la console :

- La grille
- Le score
- Le bouton d'arrêt de jeu
- Les blocs posables pour ce tour

-`trim_matrix(self, forbidden):`

Enlève l'excédent de forbidden, pour avoir une matrice de forme rectangulaire. Il s'est montré utile pour la vérification de validité de position d'un bloc qui s'apprête à être posé.

-`set_ended(self, bool):`

Arrête ou non la partie et demande au joueur s'il veut la sauvegarder. Si oui, il lui demande son nom et met toutes les variables dans `snapshots` qu'il sauvegarde ensuite dans un fichier unique avec le plateau de jeu.

Choix structuraux

Nous avons décidé de créer une classe Game contenant l'entièreté des variables qui concernerait le jeu en lui-même (en opposition avec les variables globales). Cela nous permet :

- De contenir nos variables de jeu dans un endroit précis.
- De contenir certaines méthodes propres aux parties elles-mêmes.
- D'organiser le projet dans son ensemble autour d'un objet central.

Nos difficultés techniques

- Travail d'équipe et organisation sur GitHub, mauvaise organisation au niveau des commits. Réglé par une meilleure communication.
- Divers erreurs sémantiques à cause de la complexité de certaines méthodes utilisées (`trim_matrix()` par exemple)
- Afficher la `grid` dans la console ne suffit pas, il faut aussi rajouter les coordonnées et donc la décaler pour que chaque nombre corresponde à une colonne. Or les nombres étaient mal alignés car ceux ayant deux caractères prenaient plus de place. Nous avons dû mettre une condition pour les espaces.

Bilan

Pour ce projet nous avons été amenés à utiliser des outils tels que GitHub (donc git), et CodeWithMe de JetBrains, qui permet d'ouvrir un espace collaboratif de codage en temps réel. Ils nous ont permis de mieux organiser nos tâches et de pouvoir pleinement utiliser le potentiel d'un travail groupé.

L'utilisation d'une classe a vite été le centre de notre projet, elle a été primordiale dans la structure du code et la gestion des interactions entre les différentes méthodes. Bien que non vu en cours, cette technologie nous a permis, après un petit temps d'adaptation, de créer un code solide, qui peut aisément être utilisé pour créer plusieurs instances de jeu.

Enfin, l'annotation systématique de toute nouvelle fonctionnalité et les descriptions des commits git se sont peu à peu imposées chez nous. En effet nous perdions un temps non négligeable à relire et essayer de comprendre le travail des autres.

En conclusion, ce projet en groupe a fait évoluer notre travail en équipe et plus généralement, nous a permis de mieux organiser notre travail au travers d'outils, de fonctions et de classes.