Projet Sokoban
— Étape 1 —

1 Objectifs

Dans la première étape du projet vous devez créer l'ouverture du jeu et dessiner les murs et destinations du premier niveau.

2 Les modules

Pour démarrer le projet, créez un dossier pour votre projet et à l'intérieur de celui-ci les fichiers :
— `__init__.py` : contiendra la documentation du projet.
— `__main__.py` : contiendra la fonction principale du programme (module principal).
— `avatar.py` : contiendra l'implémentation de l'avatar du joueur.
— `caisse.py` : contiendra l'implémentation d'une caisse.
— `couleur.py` : contiendra les définitions de toutes les couleurs utilisées dans le programme.
— `ouverture.py` : contiendra l'implémentation de l'ouverture du jeu.
— `scenario.py` : contiendra l'implémentation des scénarios (niveaux du jeu).

3 Documentation

Dans le ficher de documentation (`__init__.py`), vous devez faire la documentation du jeu. Cette documentation doit obéir au style utilisé dans les exemples donnés en cours (téléchargeables sur Moodle). La documentation doit contenir :
— Titre du jeu (vous pouvez créer un nouveau titre pour votre jeu) ;
— Noms des auteurs et leurs adresses email.
— Date de création.
— Discipline (Algorithme et programmation 1).
— Institution (Université d'Artois).
— Description du jeu :
  — Objectif (pour gagner).
  — Règles du jeu et le contrôle du personnage.

4 Module Couleur

Comme dans les exemples du cours, ce module va servir à déclarer toutes les couleurs utilisées pour le jeu. Pour le moment, nous en déclarerons uniquement quelques unes. Ces couleurs sont données à titre d'exemple, vous pouvez les changer si vous voulez.

```
BLANC = (255,255,255)     # pour écrire du texte
NOIR = (0,0,0)            # pour le fond de la surface
MARRON = (192, 128,  64)  # pour les murs
GRIS = (192, 192, 192)    # pour une case destination
```

5 Module Ouverture

Comme la plupart des autres modules, ce module nécessite l'import du module `pygame` (à faire tout en haut du fichier). Il aura besoin également du module `couleur` que vous venez de créer.

L'ouverture du jeu est faite avec deux fonctions. La fonction `execute` exécute l'ouverture en utilisant la fonction `dessine` qui sert uniquement à la dessiner.

Exécute : Dans le module d'ouverture (`ouverture.py`), créez la fonction `execute` qui exécutera l'ouverture du jeu. La fonction reçoit la surface comme argument et retourne `True` si le joueur a décidé de commencer le jeu et `False` sinon. La fonction doit :

1. Dessiner l'ouverture du jeu (par appel à la fonction `dessine` décrite ci-dessous)
2. Mettre à jour l'écran
3. Tant que l'ouverture n'est pas terminée :
   (a) Traite les événements de l'utilisateur (par la boucle habituelle) :
   (b) Si le joueur clique sur quitter (`pygame.QUIT`), retourner `False`.
   (c) Si le joueur appuie sur ESC (`pygame.K_ESCAPE`), retourner `False`.
   (d) Si le joueur appuie sur ENTRER (`pygame.K_RETURN`), retourner `True`.

Dessine : Dans le module d'ouverture, créez la fonction `dessine` qui dessine l'ouverture du jeu. La fonction reçoit la surface comme argument et n'a pas de valeur de retour. La fonction doit dessiner l'écran d'ouverture du jeu. Cet écran doit contenir :

— Titre du jeu en grandes lettres.
— Date de création du jeu.
— Noms des auteurs.
— Discipline (Algorithme et programmation 1).
— Institution (Université d'Artois) ; item Instructions pour le joueur (ex. : ENTRER pour commencer, ESC pour quitter, etc.).
— (**Optionnel**) Vous pouvez afficher un dessin ou une photo pour que l'ouverture du jeu soit encore plus belle !

6 Module principal

Fonction principale : Dans le module principal (`__main__.py`), écrivez la fonction principale `main` qui ne reçoit pas d'arguments et qui n'a pas de valeur de retour. C'est cette fonction qui sera appelée pour lancer le jeu. Sa structure initiale est donnée ci-dessous. Pour l'instant elle est très simple, mais cette structure évoluera par la suite, au fur et à mesure que nous ajouterons les autres éléments du jeu dans notre programme.

1. Initialise Pygame.
2. Ouvre la fenêtre de l'application.
3. Exécute l'ouverture du jeu (utilise le module `ouverture.py`).
4. Si l'utilisateur a décidé de quitter le programme :
   (a) Quitte.
5. Quitte Pygame.

6.1 Test

Testez votre programme maintenant. Il doit afficher l'écran d'ouverture et attendre l'action du joueur. Pour l'instant, il s'arrête dès que le joueur clique sur quitter ou appuie sur ESC ou ENTRER.

7 Scénario

Un scénario correspond à un niveau du jeu. Il contiendra l'avatar du joueur, les caisses, les murs et les destinations. Pour cette étape, nous allons uniquement dessiner (et pas jouer) le premier niveau du jeu.

Dans un premier temps, nous allons créer les niveaux de jeu en dur. Voici les dessins qu'on cherche à représenter :

```
#######
#@$  .#
#######
```

pour le niveau 0, avec la légende suivante :

- `#` : mur
- `$` : caisse
- `.` : destination
- `@` : personnage

Et pour le niveau 1 :

```
    #####
    #   #
    #$  #
  ###  $##
  #  $ $ #
### # ## #   ######
#   # ## #####  ..#
# $  $          ..#
##### ### #@##  ..#
    #     #########
    #######
```

Plus tard (à la fin du projet), vous pourrez créer autant de niveaux que vous voulez pour votre jeu.

Les grilles : Dans le module scénario (`scenario.py`), nous allons créer les niveaux précédents sous la forme de listes de chaînes de caractères (une chaîne de caractères par ligne de la grille du niveau) :

```
NIVEAU0 = ["#######", "#@$  .#", "#######"]
NIVEAU1 = ["    #####          ", "    #   #          ", "    #$  #          ",
           "  ###  $##         ", "  #  $ $ #         ", "### # ## #   ######",
           "#   # ## #####  ..#", "# $  $          ..#", "##### ### #@##  ..#",
           "    #     #########", "    #######        "]
```

Créez maintenant une liste `GRILLES` qui contient les deux niveaux.

Initialisation : Dans le module scénario, écrivez la fonction `init` qui reçoit un entier `num` en argument. Ceci correspond au numéro de la grille choisie. La fonction doit récupérer le niveau qui correspond à `num` dans la liste `GRILLES`. C'est-à-dire, si la fonction reçoit 0, alors elle doit lire le premier niveau, si elle reçoit 1 elle doit lire le second et ainsi de suite.

Durant la lecture du fichier, la fonction doit initialiser le dictionnaire des attributs du scénario et ensuite le retourner. Les attributs à initialiser sont :

— `joueur` : valeur initiale : `None`. Contiendra l'avatar du joueur (défini plus tard).
— `caisses` : valeur initiale : `[]`. Contiendra la liste de caisses du scénario (définie plus tard).
— `grille` : Contiendra une liste de **listes de caractères** (attention, pas une liste de chaîne de caractères). Le caractère (i,j) de la grille doit correspondre à un objet immobile du scénario.

En effet, la fonction doit parcourir le niveau et remplir la grille avec les murs et les destinations (qui sont immobiles dans le jeu). Les caisses et l'avatar du joueur (qui sont mobiles) seront représentés autrement (plus tard).

Dessin : On arrive à la partie graphique à proprement parler. Les éléments du jeu prennent tous la place d'un petit carré, que ce soit une caisse, une brique de mur, l'avatar du joeur, etc. Commencez par déclarer une constante pour la taille de ces carrés :

```
T_GRILLE = 30
```

— Écrivez la fonction `dessine_mur` qui prend en argument la surface de dessin, un numéro de ligne `l` et un numéro de colonne `c`. Cette fonction dessine une brique marron sous la forme d'un carré de dimensions `T_GRILLE x T_GRILLE` aux coordonnées `(l,c)` ;
— Écrivez la fonction `dessine_cible` qui prend en argument la surface de dessin, un numéro de ligne `l` et un numéro de colonne `c`. Cette fonction dessine une cible grise sous la forme d'un carré de dimensions `T_GRILLE x T_GRILLE` aux coordonnées `(l,c)` ;
— Écrivez la fonction `dessine` qui reçoit un dictionnaire d'attributs d'un scénario et la surface de dessin. La fonction doit effacer l'écran et ensuite dessiner tous les objets du scénario, c'est-à-dire, pour l'instant, uniquement les murs et les destinations (certains éléments seront implémentés plus tard).

Exécution : Encore dans le module scénario, créez la fonction `execute` qui reçoit un dictionnaire d'attributs d'un scénario. La fonction doit :

1. Initialiser l'horloge.
2. Tant que le niveau n'est pas terminé :
   (a) Traite les événements de l'utilisateur.
      i. Si le joueur clique sur quitter (`pygame.QUIT`), retourner `0`.
      ii. Si le joueur appuie sur ESC (`pygame.K_ESCAPE`), retourner `0`.
      iii. Si le joueur appuie sur R (`pyghame.K_r`), retourner `1` (recommencer !)
      iv. Si le joueur appuie sur ENTRER (`pygame.K_RETURN`), retourner `2`.
   (b) Effacer l'écran.
   (c) Dessiner le scénario (appel à la fonction `dessine`).
   (d) Mettre à jour l'écran.
   (e) Ajuster l'horloge.

Ajouts dans le module principal : Dans le module principal, importez le module scénario. Après l'exécution de l'ouverture du jeu et juste avant quitter Pygame, ajoutez la boucle principale du programme :

1. Tant que le programme n'est pas terminé :
   (a) Initialise le scénario du jeu (au niveau 0).
   (b) Exécute le scénario.
   (c) Si l'utilisateur a décidé de quitter, alors quitte.
   (d) Si l'utilisateur a décidé de recommencer le niveau, alors recommence.
   (e) Si l'utilisateur a décidé de continuer, alors passe au prochain niveau.

7.1 Test

Testez votre programme maintenant. Après l'ouverture, il doit afficher le premier niveau du jeu. Le joueur peut sortir en cliquant sur quitter ou avec ESC. Il peut recommencer en saisissant R. Il peut aussi passer au prochain niveau avec ENTRER.