ZAOUN Eya
COSTANTIN Perline

PeiP 1
Année 2022-2023

Rapport du projet Python
Le jeu des allumettes

1. Introduction :

Dans le cadre de notre première année en PeiP 1 à Polytech'Nice-Sophia, nous avons eu pour tâche la réalisation d'un projet informatique. Notre objectif était de réaliser le jeu des allumettes en utilisant le langage Python puis de concevoir une interface graphique à l'aide du module Turtle. Dans ce compte-rendu, nous allons exposer le déroulement de notre projet du début à la fin, en suivant le plan ci-dessous :

Table des matières :

2. Jeu des allumettes :

- *Principe et règles :*

Le jeu des allumettes est un jeu de stratégie où s'affrontent deux joueurs. Il y a plusieurs variantes de ce jeu mais les règles sont assez simples : on dispose au départ de n allumettes (ou de n tas contenant chacun un nombre d'allumette différent) et chaque joueur modifie à tour de rôle un ou plusieurs tas selon les règles.

Dans notre cas, le jeu se joue en solitaire (le joueur joue contre l'ordinateur) et voici les règles :

- Le nombre d'allumettes de départ est tiré aléatoirement par l'ordinateur.
- Le joueur est le premier à jouer.
- A chaque tour, le joueur ou l'ordinateur prend des allumettes selon une règle du jeu. Dans l'algorithme, la règle du jeu est donnée par une liste. Dans le jeu habituel, la règle sera r=[1,2,3] : ainsi, le joueur peut choisir de retirer 1, 2 ou 3 allumettes .
- Le joueur qui prend la dernière allumette a gagné. Le perdant est donc celui qui ne peut plus prendre d'allumettes (0 allumettes restantes) ou celui qui ne peut plus jouer car il reste un nombre

d'allumettes inférieur au nombre minimal qu'il est possible de prendre à chaque tour (nombre défini par la règle).

- *Choix du niveau :*

Pour réaliser ce projet, nous avions le choix entre trois niveaux de difficultés. Notre choix s'est porté sur le niveau 1 : nous n'avons, ni l'une ni l'autre, pris la spécialité NSI au lycée et débutons dans la programmation. Nous ne maîtrisions donc pas encore totalement le langage Python et avons préféré prendre le niveau le plus facile étant donné du temps dont nous disposions pour mener à bout ce projet. De plus, nous avons pensé qu'il valait mieux essayer de faire un bon projet avec un niveau facile plutôt que de se lancer dans un projet plus difficile en n'ayant ni les compétences ni le temps pour bien le finir et rendre quelque chose de non abouti.

Voici donc les règles que suit notre projet :
- On ne dispose que d'un seul tas d'allumettes au départ
- Le nombre d'allumettes de départ est tiré aléatoirement
- Les choix de l'utilisateur sont lus avec la fonction int(input))
- La règle du jeu est donnée dans une liste de la forme r=[1,2,3]

- *Stratégie gagnante :*

Le jeu des allumettes est un jeu de stratégie pure, et nous avons trouvé intéressant d'exposer la conjecture d'une stratégie gagnante, même si nous ne l'utilisons pas dans notre code (le joueur peut bien sûr l'utiliser mais elle n'est pas donnée pour orienter les choix de l'ordinateur) :

Exemple avec 4 allumettes :
• si le premier prend 1 allumette alors l'autre en prend 3 et gagne
• si le premier prend 2 allumettes alors l'autre en prend 2 et gagne
• si le premier prend 3 allumettes alors l'autre en prend 1 et gagne
On en déduit que si l'on joue avec 4 allumettes, le premier joueur a perdu

Voici donc un tableau qui résume le choix à faire selon le nombre d'allumette restant :

Allumettes restantes	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**…**
Allumettes à prendre pour gagner	1	2	3	perdu	1	2	3	perdu	1	2	3	perdu	1	2	…

La stratégie dépend donc du nombre d'allumettes restantes et du joueur qui commence : pour être sûr de gagner à tous les coups, il faut laisser l'adversaire commencer si le nombre initial d'allumettes est un multiple de 4. Ensuite, il s'agit d'amener l'adversaire à 4 allumettes. Il faut donc, à notre tour, laisser un nombre d'allumettes correspondant à un multiple de 4, c'est à dire 4, 8, 12, 16, 20…
Si notre adversaire connaît la stratégie gagnante, on ne peut rien faire, à part espérer une erreur de sa part. S'il ne la connaît pas, il joue au hasard et finira sûrement par se tromper. On peut aussi choisir de retirer à chaque fois une seule allumette pour faire durer la partie et augmenter les risques d'erreur de l'adversaire.

3. Interface :

- *Choix du thème :*

Nous devions représenter les allumettes par un dessin fait avec le module Turtle, et les intégrer dans un décor, en laissant parler notre créativité pour choisir le thème. Nous avons choisi comme thème la plage et les voiliers. L'idée était de faire un décor assez estival, coloré, tropical et enfantin pour apporter un peu de joie et de chaleur dans cette période hivernale !

- *Description de différentes zones :*

Le décor inclut les éléments suivants :

- Une zone pour afficher ce qui représente les allumettes : la mer
- Les allumettes en jeu : bateau avec des voiles
- Les allumettes quand elles ont été retirées : bateau sans voiles (seulement la coque)
- Un élément du décor à deux endroits différents : les palmiers sur la plage
- Un élément du décor (différent des allumettes) répétitif : les étoiles de mer/ les poissons
- Une zone permettant l'affichage textuel du nombre courant d'allumettes : le soleil
- Une zone permettant l'affichage textuel du nombre d'allumettes retirées à chaque coup : un nuage

4. Structure du projet :

- *Codage en Python du jeu :*

Le programme principal pour le jeu suit l'algorithme suivant :

```
Tirer un nombre aléatoire d'allumettes
Tant qu'il n'y a pas de perdant :
    Afficher les allumettes
    Faire jouer le joueur
    Si le joueur ne peut pas jouer : le joueur a perdu
    Sinon
        Afficher les allumettes
        Faire jouer l'ordinateur
        Si l'ordinateur ne peut pas jouer, l'ordinateur a perdu
```

Voilà comment se décompose notre code pour le jeu :

Nous avons tout d'abord créé deux fonctions principales : la première pour le tour du joueur et la seconde lorsque c'est à l'ordinateur de jouer.
Pour la fonction du joueur, le nombre d'allumettes à enlever est demandé avec la fonction int(input()) et dans l'interface, avec la fonction int(numinput()) qui permet d'ouvrir une fenêtre pour interagir avec le joueur directement sur le décor.
Pour la fonction de l'ordinateur, le choix du nombre d'allumettes à retirer se fait avec la fonction choice(regle) qui permet de tirer un nombre aléatoire dans la liste qu'est la règle. Nous avons précédemment importé le module random (from random import)pour pouvoir utiliser cette fonction.
Selon les nombres donnés par chacun des joueurs, nous avons étudié différentes possibilités dans les fonctions :
Si le nombre n'est pas dans la règle, on demande à l'utilisateur de rentrer à nouveau un nombre qui se trouve dans la liste (règle). Pour vérifier cela, nous avons utilisé la fonction in().
Si le nombre est inférieur au nombre d'allumettes restantes, on demande de donner un nombre plus petit
On redemande donc un nombre tant que le nombre donné n'est pas dans la règle ou est trop grand.
Après chaque tour, le nombre d'allumettes restantes se met à jour.

Puis nous avons écrit le corps du jeu, en dehors de nos fonctions :
Premièrement les paramètres du jeu sont définis : on entre une liste pour le règle, on tire un nombre aléatoire d'allumettes de départ avec la fonction randint(min, max)… Sur le décor, on dessine le nombre d'allumettes initial avec un appel à la fonction pour dessiner les allumettes.
Pour pouvoir jouer tant qu'il n'y a pas de gagnant (while not gagnant), nous avons initialisé une variable « gagnant » à False et lorsque le nombre d'allumettes vaut 0, cette variable prend la valeur True. C'est alors la fin de la partie.
Lorsque c'est au tour du joueur, la fonction correspondante (tourJoueur) est appelée et lorsque c'est à l'ordinateur, on fait appel à la fonction pour l'ordinateur (tourOrdinateur).

- *Développement de l'interface graphique :*

En premier lieu, nous nous sommes mises d'accord sur le thème et nous avons fat la maquette version papier de notre interface en plaçant les différentes zones.

Nous avons ensuite réfléchi à la façon dont décomposer et découper les différents éléments à l'aide de formes simples.
Après avoir créé toutes nos fonctions pour les formes élémentaires (dessineRectangle, dessineDemiCercle,dessineFeuilles…), nous les avons combinées afin de faire les différents objets (dessinePalmier, dessineBateau…).

Notre code pour l'interface graphique se décompose en deux fonctions principales : une qui dessine le décor (tous les éléments à leur place) et l'autre pour dessiner nos allumettes (bateaux). Pour changer la forme des allumettes sans modifier le décor, nous avons utilisé 2 tortues du module Turtle (tk pour le décor et tc pour les allumettes) et la fonction clear().

Pour ne pas fusionner notre partie avec le code du jeu et celle qui dessine le décor, nous gardé deux fichiers séparés et enregistrés dans le même dossier et nous avons importé le décor au début de notre code pour le jeu (from decor import *), de façon à rendre le tout plus lisible et moins chargé.

Voici les fonctions déjà existantes de turtle que nous avons utilisé et intégré dans nos propres fonctions et procédures :
- **tracer (1,0)** : pour tracer rapidement tout le décor sans voir les étapes et les animations de la tortue
- **width()** : pour changer l'épaisseur du trait de dessin
- **bgcolor()** : pour changer la couleur du fond de l'écran
- **fillcolor() / begin_fill() / end_fill()** : pour colorier l'intérieur des formes
- **write()** : pour écrire des messages
- **int(numinput())** : pour interagir avec l'utilisateur avec une fenêtre pop-up et pouvoir entrer des nombres
- **setheading()** : pour réorienter la tortue
- **circle(randint(1,8))** : pour faire des cercles avec des rayons choisis aléatoirement entre 1 et 8
- **x=randint(min,max**) : pour choisir une position aléatoire pour x entre un minimum et un maximum (même chose avec y)
- **colormode(255**) : pour changer le mode d'entrée de couleurs et pouvoir saisir des couleurs en RGB (indispensable pour la fonction suivante)
- **r= randint(0,255) / g= randint(0,255) / b = randint(0,255)** : pour générer aléatoirement une couleur (pour cela, on génère aléatoirement une nuance de rouge, de vert et de bleu) -> utilisé dans notre fonction random_color()
- **time.sleep(2)** : pour attendre 2 secondes avant de faire l'action suivante
- **clear()** : pour effacer les éléments tracés par une certaine tortue.

Nous avons dans un premier temps créé les fonctions pour créer les zones et les éléments obligatoires puis nous en avons rajouté pour que notre décor soit plus riche (poissons, nuages, serviettes, écume).

5. Répartition du travail :

La bonne répartition du travail était un point fondamental compte tenu du temps imposé pour finir le projet : nous nous sommes donc mises d'accord pour accomplir chacune certaines tâches et travailler individuellement sur différents programmes avant de les regrouper :

- **Code du jeu :** ensemble durant les séances avant les vacances
- **Maquette :** Perline a fait un dessin sur papier et Eya en a fait une version numérique
- **Fonctions pour les formes élémentaires et assemblage des formes pour créer les différents éléments du décor :** Perline
- **Arrière-plan et calcul de la distance entre les bateaux :** Eya
- **Mise en commun des différents programmes et dernières modifications :** toutes les deux à la rentrée
- **Derniers éléments du décor (poissons et serviettes) :** Eya
- **Ecriture des commentaires :** Perline

- **Rédaction du rapport :** au fur et à mesure toutes les deux (document partagé)

6. Difficultés rencontrées et solutions :

Au départ, nous avions fait le code du jeu sans de def de fonctions, puis nous avons compris notre erreur et avons défini des fonctions pour que le programme soit plus général et concis.

Nous avons ensuite rencontré des difficultés vis-à-vis de la fonction pour dessiner les allumettes cassées ou non et pour connaître la position des bateaux mais au bout de plusieurs tentatives, nous avons résolu ces problèmes en intégrant dans notre fonction pour dessiner les bateaux un booléen (cassé) prenant la valeur True ou False selon si le bateau a été retiré ou non. En ce qui concerne la fonction dessineAllumettes, nous avons ajouté dans les paramètres une variable qui stocke le nombre total d'allumettes et une autre pour le nombre restant : ainsi, on appelle la fonction pour dessiner tous les bateaux, et pendant qu'on dessine, on teste si c'est le moment de dessiner des bateaux cassé (par rapport au nombre total d'allumettes et le nombre restant).
Pour la position de ces bateaux, voici comment nous avons raisonné : nous avons fait une boucle pour dessiner les 7 premiers bateaux sur une même ligne. On donne une position initiale à x et on ajoute à chaque fois le nombre de bateaux affichés multiplié par la longueur (l) d'un bateau (ex : pour le premier x + 0 * l = x, pour le deuxième x + 1 * l etc). On teste ensuite directement quand on appelle la fonction si on a dépassé le nombre de bateaux pas cassés et si c'est le cas, on enlève 7 à i à partir du 8ème bateau.

De plus, au début, nous avions utilisé des boucles while pour dessiner les décors répétitifs sur une seule et même ligne (ex : les étoiles de mer) mais nous nous sommes rendu compte qu'il était plus simple d'utiliser des « for x in range (start, stop, step) » pour changer la position de x de façon régulière.

Nous voulions pouvoir jouer directement sur le décor plutôt que dans le script mais nous ne savions pas comment faire. Nous avons découvert par la suite les fonction write() et int(num(input()) et nous avons pu les intégrer à nos fonctions existantes.

Nous avons ensuite remarqué une utilisation répétée des 3 fonctions up(), goto(x,y) et down(), nous avons alors écrit une fonction (déplacer) pour éviter de devoir écrire souvent ces trois lignes

7. Améliorations possibles:

Un projet comme celui-ci pourra toujours être amélioré et c'est d'ailleurs un peu frustrant (ce n'est jamais parfaitement fini). Pour améliorer notre projet, nous aurions pu rajouter des formes de base combinées pour créer de nouveaux éléments de décor plus complexes.
Nous aurions également pu rajouter certains détails comme des ombres, des objets en 3D comme l'écume que nous avons rajoutée à la fin.

8. Conclusion :

A l'aide de ce projet nous avons pu comprendre et expérimenter les différentes étapes de la conception d'un jeu informatique, en commençant par la création d'un programme de jeu, puis d'une interface, et en combinant ensuite les deux pour obtenir un rendu agréable et visuel. D'autre part, la programmation nous a permis d'améliorer et de consolider nos connaissances en langage Python. En plus d'être un projet pédagogique, il est aussi ludique et nous a donné beaucoup de liberté dans le code et dans la conception de l'interface graphique. Nous avons particulièrement aimé toute la partie de conception du décor, qui a stimulé notre créativité.

Nous finissons ce projet avec la satisfaction et la fierté de l'avoir réussi par nous-même, de A à Z, alors que nous le trouvions au départ très compliqué !