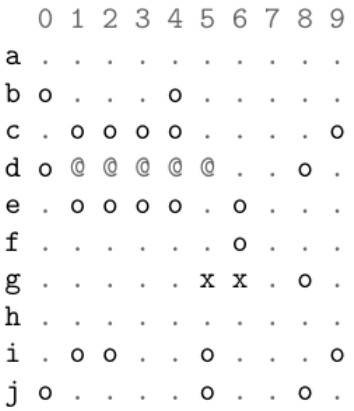
L2 MIASHS - Informatique S3
Bataille navale

Garance Gourdel

2021-2022

Le but des prochaines séances est de réaliser un petit projet de bataille navale en console. il faut pouvoir placer les bateaux, donner une position de tir, jouer contre l'ordinateur ou contre un autre joueur en local (sur le même ordinateur).



C'est un projet à réaliser en groupe de 2 idéalement, 3 si besoin. Vous devrez rendre votre projet et il sera évalué, à la fois par une relecture de votre code et par une petite soutenance.

Le projet est raisonnablement long (moins de 300 lignes de code) mais il est important que vous le travaillez en dehors des TDs. Les TDS servirons à vous débloquer si vous coincez. Le sujet vous demande d'écrire des fonctions distinctes pour chaque élément important, testez-les soigneusement et maintenez votre code le plus clair possible. Pensez également à faire des sauvegardes régulières.

Ndt: Si vous êtes familier.e avez un gestionnaire de version comme git, leur utilisation est très utile dans des projets comme celui ci.

Vous pouvez commencer par copier-coller ces constantes en haut de votre fichier projet. Elles nous seront utiles tout au long du sujet. Ndt: leurs noms sont tout en majuscules par convention car ce sont des variables globales constantes.

```
#Constantes
N=10
COLONNES=[str(i) for i in range(N)]
LIGNES = [' '] +  list(map(chr, range(97, 107)))
DICT_LIGNES_INT = {LIGNES[i]:i-1 for i in range(len(LIGNES))}
VIDE = '.'
EAU='o'
TOUCHE='x'
BATEAU='#'
DETRUIT='@'
NOMS=['Transporteur','Cuirassé','Croiseur','Sous-marin','Destructeur']
TAILLES=[5,4,3,3,2]
```

Nous allons avoir deux structures principales pour un joueur:

- Une matrice de caractères `m` de taille $N \times N$ (Rappel, on a définit $N = 10$ dans les constantes). Cette matrice représente la carte où l'on notera les tirs effectués par l'adversaire. Une case sera soit `VIDE` si la case n'a pas été attaquée, `EAU` si la case a été attaqué et qu'il n'y a pas de bateau. `TOUCHE` si la case à été touchée et le bateau à cette position n'a pas encore été détruit. Et finalement `DETRUIT` si la case a été touchée et le bateau dessus a été coulé.
- Une liste de bateaux `flotte` qui représentera tous nos bateaux qui n'ont pas encore été coulés. La façon dont les bateaux seront définit sera expliqué dans la Section 2.

1 Carte et tir

Cette section à pour but de créer la matrice représentant la carte, de pouvoir l'afficher et de créer une fonction de tir qui met à jour la carte pour marquer quels positions ont été attaquées.

1. Écrivez une fonction `create_grid` qui ne prends aucun arguments et renvoi une matrice de taille $N \times N$ remplie avec le symbole `VIDE`.
2. Écrivez une fonction `plot_grid` qui prends en argument une matrice `m` de taille $N \times N$ et l'affiche avec le numéro des lignes et des colonnes (définit dans les constantes `LIGNES` et `COLONNES`). Ndt: pour éviter que l'affichage soit lent, il vaut mieux construire la chaîne de caractère à afficher avec des concaténation et l'afficher à la fin de la fonction, plutôt que de faire plein de petit `print` qui peuvent ralentir votre programme. Le résultat devrait ressembler à ceci:

```
  0 1 2 3 4 5 6 7 8 9
a . . . . . . . . . .
b . . . . . . . . . .
c . . . . . . . . . .
d . . . . . . . . . .
e . . . . . . . . . .
f . . . . . . . . . .
g . . . . . . . . . .
h . . . . . . . . . .
i . . . . . . . . . .
j . . . . . . . . . .
```

Testez votre programme.

3. Écrivez une fonction `tir` qui prends deux arguments: une matrice `m`et une pair d'entiers entre 0 et $N-1$ `pos`, et remplace la valeur de case `pos`de matrice `m`par `EAU`. Par exemple un `tir(m,(2,3)` suivi de `plot_grid(m)` affichera le résultat suivant:

```
  0 1 2 3 4 5 6 7 8 9
a . . . . . . . . . .
b . . . . . . . . . .
c . . . o . . . . . .
d . . . . . . . . . .
e . . . . . . . . . .
f . . . . . . . . . .
g . . . . . . . . . .
h . . . . . . . . . .
i . . . . . . . . . .
j . . . . . . . . . .
```

Testez votre programme.

4. Écrivez une fonction `random_position` qui ne prends aucun argument et renvoi une pair d'entiers entre 0 et $N-1$. Vous pourrez utilisez les fonctions `randint` ou `randrange` du module random mais n'oubliez pas de l'importer auparavant.

5. Servez vous de votre fonction `random_position` pour tirez sur des positions au hasard. Vous pourrez utiliser une boucle `while` comme celle ci pour voir le résultat:

```
map1= create_grid()
while True:
    plot_grid(map1)
    pos = random_position()
    tir(pos,map1)
    input()
```

6. Écrire une fonction `pos_from_string` qui prend en argument une chaîne de caractère S tel que `'b 4'` et le transforme en position (pair d'entier) comme `(1,4)`.

7. Testez votre programme. Qu'affiche t'il si on lui donne une chaine non conforme comme 'b 4' ou 'z 10' ? Il doit afficher un message rappelant à l'utilisateur le format et lui redemander une position.

8. Si besoin, modifiez votre programme pour qu'il affiche un message de refus si la position a déjà été attaquée et redemander une position.

2 La flotte

Le but de cette section est de gérer la création des bateaux stockés dans la liste de bateaux `flotte`. Un bateau est un dictionnaire avec la structure suivante :

- à la clef `'nom'` est associé le nom du bateau comme "Transporteur", "Cuirassé", "Croiseur" ...
- à la clef `'taille'` est associé un entier représentant la taille du bateau
- à la clef `'cases touchés'` est associé un entier: le nombre de case du bateau qui ont été touchés.
- à la clef `'positions'` est associé une liste de pairs d'entiers entre 0 et $N-1$ représentant les positions que le bateau occupe.

1. Écrivez une fonction `nouveau_bateau(flotte,nom,taille,pos,orientation)` qui ajoute un bateau à la liste de bateau `flotte`étant donné une position `pos`, une orientation ('h' ou 'v'), un nom et une taille de bateau. Testez votre code avec les instructions suivantes:

   ```
   flotte=[]
   nouveau_bateau(flotte,"Test",3,(0,0),'h')
   print(flotte)
   ```

   Le résultat devrait être : `['nom': 'Test', 'taille': 3, 'cases touchés': 0, 'positions': (0, 1), (0, 2), (0, 0)]`

2. Ecrivez une fonction `presence_bateau(pos,flotte)` qui renvoie `True` si la position `pos`est occupé par un bateau de `flotte`et `False` sinon.

3. En vous inspirant de votre fonction `plot_grid` (conservez la fonction `plot_grid`), et à l'aide de `presence_bateau` créez une nouvelle fonction `plot_flotte_grid(m,flotte)` qui affiche le caractère `BATEAU` aux positions qui contiennent un bateau.

4. Écrivez une fonction `input_ajout_bateau(flotte,nom,taille)` qui demande à l'utilisateur (grâce à la fonction `input`) une position comme 'a 0' ou 'f 7' et une orientation 'h' ou 'v' pour ajouter le bateau `nom` et qui grand de `taille` cases à `flotte`.

5. Que se passe t'il si l'utilisateur ne rentre pas une position valide comme `'hfiuhz'`, `'g 3 h'` ? Votre programme ne doit pas planter, il doit rappeler le format à l'utilisateur et lui redemander une position. Idem pour l'orientation.

6. Essayez d'ajouter un bateau qui dépasserait de la carte comme 'a 9' 'h' pour un bateau de taille 3. Que se passe t'il ? Le comportement attendu est que le programme n'ajoute pas le bateau, et explique à l'utilisateur le problème. Vous pouvez modifier votre fonction `nouveau_bateau` pour qu'elle essai d'ajouter le bateau à la position donné et renvoi `False` si ce n'est pas possible et `True` si le bateau a été correctement ajouté.

7. Que se passe t'il si vous essayez d'ajouter un bateau à une position déjà prise par un autre bateau ? De la même façon que pour la question précédente le bateau ne doit pas être ajouté et il faut redemander à l'utilisateur une position.

8. Écrivez une fonction `init_joueur()` qui demande à l'utilisateur de placer tous les bateaux définit dans les constantes `NOMS` et `TAILLES`, et renvoi une carte vide `m`et une la flotte choisi par l'utilisateur `flotte`.

9. Écrivez une fonction `init_ia()` qui tire des positions au hasard ajoute tous les bateaux définit dans les constantes `NOMS` et `TAILLES`, et renvoi une carte vide `m`et une la flotte créer au hasard : `flotte`. Il se peut que la position choisie au hasard ne convienne pas (dépassement hors de la carte, collision avec un autre bateau), dans ce cas il faut continuer à tirer des positions au hasard jusqu'à en trouver une qui convient.

3 Touché coulé

Le but de cette section est de mettre à jour votre fonction `tir` pour qu'elle fonctionne comme dans un jeu de bataille navale classique.

1. Rajoutez `flotte`comme argument de votre fonction `tir`, la fonction doit être `tir(pos,m,flotte)` et renvoyer `True` si on a pu tirer sur `pos`et `False` si la position avait déjà été attaqué.

2. En utilisant la fonction `presence_bateau` définit auparavant, modifiez `tir` pour qu'elle déclare si une position est "Manquée !" ou "Touché !" et change la valeur de la case à `TOUCHE`.

3. Écrivez une fonction `id_bateau_at_pos(pos,flotte)` qui vous renvoi l'indice (la position dans la liste) du bateau à la positon `pos`s'il y en a un, `None` sinon.

4. Modifiez `tir` pour qu'elle mette a jour la valeur associé à `'cases touchés'` si un bateau est touché.

5. Modifiez `tir` pour qu'elle vérifie si un bateau est "touché-coulé!" et l'annonce en donnant son nom. Puis la fonction doit retirer le bateau de la flotte grâce à `flotte.pop(i)` qui permet d'enlever l'élément en position `i` et changer la valeur de toutes les cases du bateau à `DETRUIT`.

6. Testez votre fonction, vous pourrez utiliser une initialisation de carte et flotte au hasard grâce à `init_ia` et des tirs au hasard en utilisant `random_positions`.

4 Intelligence artificielle et deux joueurs

Le but de cette section est d'avoir un petite "intelligence" artificielle pour rendre le jeu jouable à un joueur. Les tirs de l'IA seront dans un premier temps choisit complètement au hasard et dans un second temps ils seront choisit au hasard jusqu'à avoir trouvé une position "Touchée !" au quel cas l'IA visera les cases adjacentes jusqu'à avoir détruit le bateau.

1. Écrivez une fonction `tour_ia_random(m,flotte)` qui tire une position au hasard jusqu'à en trouver une qui n'a pas déjà été attaqué dans `m`et attaque cette position.

2. Écrivez une fonction `tour_joueur(nom,m,flotte)` qui demande a l'utilisateur une position sur laquelle tirer (comme `'c 5'`) jusqu'à ce que l'utilisateur en donne une correcte et qui n'a pas déjà été attaquée et tire sur cette position.

3. Écrivez une fonction `tour_ia_better_random(m,flotte)` qui regarde s'il y a un bateau touché et non détruit dans `m`. S'il y en a une, il tire sur une case adjacente qui n'a pas encore été attaquée, sinon il tire au hasard.

4. Écrivez une fonction `test_fin_partie(nom,m,flotte,nb_tour)` qui teste si la flotte est vide (ce qui signifie une victoire), affiche qui a gagné et en combien de tours, puis termine le programme grâce à la fonction `exit()`.

5. Combinez les fonctions `init_ia`, `init_joueur`, `tour_joueur`, `tour_ia` et `test_fin_partie` pour écrire une fonction `joueur_vs_ia()` qui joue les tour d'une IA et d'un joueur humain jusqu'à ce que l'un gagne.

6. Écrivez une fonction `hide` contenant de nombreux `print` pour nettoyer l'écran afin de pouvoir le passer à l'autre joueur. Combinez les fonction `init_joueur`, `tour_joueur`, `hide` et `test_fin_partie` pour créer une fonction `deux_joueurs()` qui joue une partie entre deux joueurs.

7. Modifiez votre programme pour donner le choix à l'utilisateur de jouer à 1 ou 2 joueurs ! Vous pouvez également ajouter la possibilité de choisir son nom.

5 À vous de jouer!

Vous avez programmez un petit jeu et vous êtes maintenant libre de l'améliorer et le modifiez comme vous le souhaitez (si vos modifications sont lourdes pensez à sauvegardez le projet auparavant pour pouvoir le rendre également). Voilà quelques idées d'améliorations optionnelles et non guidées :

1. Boulet de canon une fois dans la partie qui attaque les 9 cases autour.
2. Deux tirs par tour autorisé au lieu d'un.
3. Amélioration de l'IA
4. Modifications de l'affichage, avec des couleurs par exemple.

Mais tout autre idée est la bienvenue, place à l'imagination !