

Rapport du jeu akiba.

Marin Dorange Launey
L1 info Gr 4B
21601861

21 avril 2017

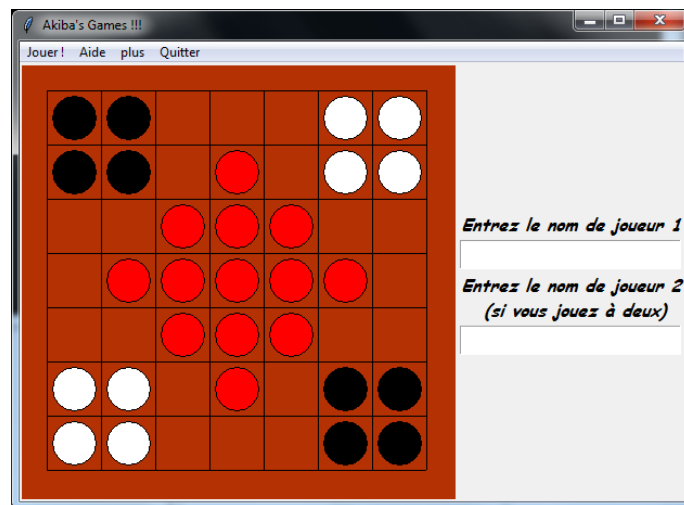


FIGURE 1 –

Table des matières

1	Introduction	3
2	Aspect esthétique	3
3	Aspect technique	3
3.1	Fonctionnalité de base	3
3.1.1	initialisation d'une partie en ligne de commande	3
3.1.2	les mouvements	4
3.1.3	les test des mouvements	6
3.1.4	la gestion des points	6
3.2	Fonctionnalité rajouter	7
3.2.1	Une interface graphique	7
3.2.2	Une IA	7
3.2.3	Un enregistrement,affichage des scores réalisé.	8
4	Suite voulue	8
4.1	Une IA plus intelligente	8
4.2	Une meilleur gestion des clic de la souris	9
4.3	Des bruitages et/ou animation	9
5	Erreur et test de fonction représentative du programme.	9
5.1	anti_deplacement_contraire	9
5.2	B	9
5.3	test_fin_partie	9
5.4	test_Pions_pousser	9
5.5	traitement_coordone	9
5.6	test_deplacement	9
5.7	compte_pions_coul	10
5.8	search_by_name	10
5.9	Grille_Initiale	10
6	conclusion	10

1 Introduction

Le jeu d'akiba est un jeu de société créé en 1994 par Serge Cahu, il est aussi connu sous le nom de Traboulet ou encore Kuba. Il s'adresse aux enfants comme aux adultes. Malgré les rumeurs ce jeu est très différents dans sa manière de jouer par rapport au célèbre jeu de société Abalone. Toutes fois si vous souhaitez acheter ce jeu de société en France vous ne pourriez pas car l'éditeur de jeu Fun Connection ne le commercialise plus à cause de ces rumeurs. Nous avons donc, dans le cadre du projet de méthodologie 2016/2017 programmer ce jeu à l'aide de Python et de quelques bibliothèques telles que Tkinter.

J'ai choisi ce jeu car il me rappelle le jeu d'Abalone malgré la grande différence, de plus ce jeu m'a plus facilement inspiré des idées que pour le Latruncule.

2 Aspect esthétique

Pour l'esthétique, j'ai voulu choisir un style plutôt rustique, neutre. Ce choix est dû au fait que le jeu inspire la sagesse, la réflexion. Pour "faire" ce style j'ai choisi que la grille pour jouer est un fond marron rappelant le bois et donc le thème principal. Ensuite même si l'on peut jouer sur les nuances, les couleurs des pions ont été définies selon les couleurs prédéfinies de Tkinter (red, black, white). La police a été choisie pour continuer sur le style rustique, neutre. Ainsi la police est noire, de type "comic sans ms". La taille varie selon que le texte soit un titre ou non. Enfin les titres sont en italique, gras et souligné. Pour finir la disposition du plateau, des menus et de l'affichage et, il me semble très communes, en effet, les menus en haut, la partie jeu (ici le plateau) à gauche et l'affichage à droite se retrouvent très souvent dans les jeux flash, etc.

3 Aspect technique

3.1 Fonctionnalité de base

3.1.1 initialisation d'une partie en ligne de commande

C'est une fonctionnalité assez simple. On demande tout d'abord à ce que l'on saisisse le nom des joueurs. Ensuite on crée une grille remplie de boules, en effet on ne peut laisser des espaces vides, sans élément, dans une grille/liste. Ainsi les espaces vides sont codés par des "boules" vides représentés par des "V", les boules noires sont représentées par des "N", les boules blanches sont représentées par des "B" et enfin les boules rouges par des "R". Ensuite tant que le jeu n'est pas fini on demande au joueur qui doit jouer un mouvement et une position pour déplacer ses boules.

3.1.2 les mouvements

la fonctionnalité des mouvement regroupe 4 fonction, en effet, j'ai découpé cette fonctionnalité en 4 fonction selon les 4 mouvement possible (Haut, Bas, Gauche, Droite). C'est fonction ont été pour moi des fonction dur à réaliser. En effet le problème qui se pose et que nous ne pouvons prédire la couleur des autres.

En effet, comme le montre l'exemple ci-dessous, si nous voulons déplacer la boule blanche, on ne peut savoir si elle entrainera une boule noir ou rouge. Ainsi la première solution qui était de mettre un rang plus loin la représentation de la boule ne marche pas. Si on analyse la situation, on modifie notre support de travail, je m'explique, on met au rang $n+1$ de plateau ce que l'on voit au rang n de plateau or si l'on renouvelle une seconde fois l'opération, on met au rang $n+2$ du plateau ce que l'on voit au rang $n+1$ or précédemment le rang $n+1$ à déjà changé de couleur.

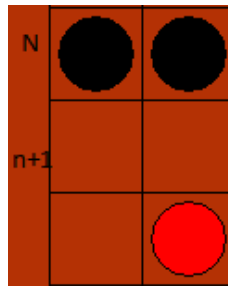


FIGURE 2 –

Ainsi sur ce morceau de plateau, on va mettre la boule noir du rang n au rang $n+1$, enfin on mettra au rang n une "boule" vide codé "V". Donc sur cette exemple tout ce passe bien. Ce mouvement effectué on se retrouve face à la boule rouge. C'est ici que, comme dit plus haut cela devient impossible d'effectuer un mouvement correcte.

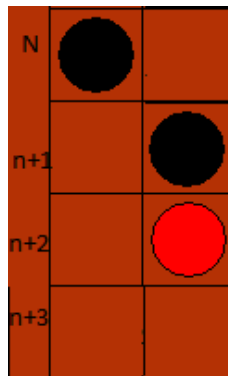


FIGURE 3 –

Ici, si on déplace le pions noir au rang $n+1$, on mets au rang $n+2$ ce que l'on voit au rang $n+1$ ici une boule noir, puis comme la case n'était pas vide on continue le mouvement on mets au rang $n+3$ ce que l'on voit au rang $n+2$. Comme la case $n+2$ à était changée juste avant ce n'est plus une boule rouge que l'on mettra mais une boule noir.

J'ai donc eu comme idée de passer par une liste représentant la colonne ou la ligne de la grille sur laquelle on travaille,

N	N	V	V	V	B	B	1
N	N	V	R	V	B	B	2
V	V	R	R	R	V	V	3
V	R	R	R	R	R	V	4
V	V	R	R	R	V	V	5
B	B	V	R	V	N	N	6
B	B	V	V	V	N	N	7
A	B	C	D	E	F	G	

FIGURE 4 –

Prenons l'exemple d'un mouvement du pions noir noté 'N' en A2 vers la gauche, la représentation de la liste sera donc :

N	N	V	R	V	B	B	
---	---	---	---	---	---	---	--

FIGURE 5 –

Par contre si on effectue un mouvement en A1 vers le bas, la représentation de la liste sera alors :

N
N
V
V
V
B
B

FIGURE 6 –

En faisant attention de copier la liste (d'où la fonction copie) et non de faire une simple égalité entre ma liste et la liste représentant la grille(dans ce cas on ne copierait que l'adresse de la liste).Ainsi on met au rang n+1 de la grille ce que l'on voit au rang n de la liste.

3.1.3 les test des mouvements

Cette fonctionnalité de base a pour but d'obliger les joueurs à respecter les règles. Ainsi je vais donc détailler les règles, la liste des contraintes.

- Une contrainte simple et première on doit tout d'abord s'assurer que le pion que l'on veut pousser soit de la bonne couleur.
- Une des règles et que le pion que l'on veut pousser ne doit avoir aucun pion derrière lui, c'est à dire que l'on peut s'imaginer mettre son pion derrière celui-ci sans risquer de toucher un autre pion.
- On ne doit pas pouvoir pousser un de ses pions en dehors du jeu.
- Enfin dernière règle, selon moi plus difficile à comprendre, on ne doit pas pouvoir annuler un coup précédemment joué par son adversaire. En effet si au tour d'avant l'adversaire a joué un coup qui a déplacé mes pions, je n'ai pas le droit de "remettre" ses pions.

Les points les plus difficiles à être, pour moi, de vérifier que l'on ne sorte pas un de ses pions en dehors du jeu ou que l'on annule un coup précédemment joué par l'adversaire. En effet pour les deux premières règles il suffit de regarder si la couleur du pion que l'on veut déplacer soit de la même couleur que celle du joueur qui doit jouer et que au rang $n+1$ ou $n-1$ se trouve une "boule" vide ou un bord du plateau.

Ensuite pour vérifier que l'on ne pousse pas un de ses pions en dehors du plateau, j'ai repris l'idée de la liste dans la fonctionnalité mouvement. En effet je parcours, lit la liste et regarde si avant d'arriver à la fin de cette liste je tombe sur une "boule" vide et si j'arrive à la fin, je regarde si la dernière boule est de la même couleur que celle du joueur qui doit jouer, dans ce cas j'empêche le programme d'aller plus loin sinon dans tous les autres cas je le laisse effectuer l'action.

Enfin si l'on veut empêcher un joueur d'annuler le mouvement réalisé par son adversaire il faut donner un peu de mémoire ou stocker les coordonnées, le mouvement effectué ainsi que la liste représentant la ligne, colonne ou il a réalisé son coup. Le choix de variable était un dictionnaire, ceci pour éviter d'avoir à retenir la position des éléments. Le problème que j'ai rencontré au départ, est que je travaillais sur la liste au coup n or je ne pouvais déterminer si précédemment il y avait un espace ou si les boules du joueur étaient déjà accolées. Pour résoudre cela j'ai ajouté la liste du joueur précédent comme indiqué plus haut, ainsi je regarde le coup $n-1$.

3.1.4 la gestion des points

Cette fonctionnalité consiste à ajouter des points au joueur si celui-ci pousse une boule rouge ou une boule adverse en dehors du plateau et à vérifier si un joueur n'a pas terminé la partie. Les points sont inscrits dans un dictionnaire ayant pour clé :

- le nom du joueur pour le nombre de boule adverse
- le nom du joueur concaténé de la lettre "R" pour le nombre de boules rouges

Pour gagner il faut vérifier une des deux conditions suivantes :

- Avoir poussé 7 boules rouges en dehors du jeu.
- Avoir poussé toutes les boules adverses en dehors du jeu.

Pour la vérification des boules poussées on s'inspire fortement de la façon dont on teste si l'on pousse une de ses boules en dehors du plateau (cf : les tests des mouvements). En effet on va parcourir la ligne ou colonne, plusieurs cas s'offrent à nous :

- Si l'on rencontre une "boule" vide on renvoie notre variable inchangée.
- Si l'on ne rencontre aucune boule vide et que la dernière boule est une boule adverse on renvoie notre variable ajoutée de un point pour le nombre de boules adverses du joueur.
- Enfin, si l'on ne rencontre aucune "boule" vide et que la dernière boule est une boule rouge on renvoie notre variable ajoutée de un point pour le nombre de boules rouges du joueur.

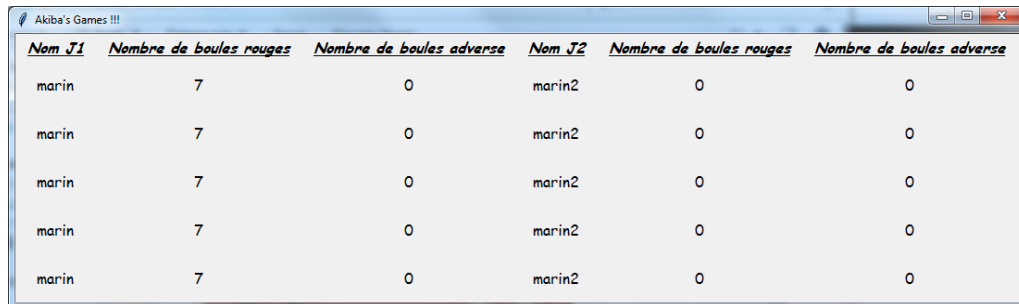
Ensuite on vérifie que les scores maximaux n'ont pas été atteints pour l'un des deux joueurs, Si c'est le cas on affiche alors un message pour l'avertir et on l'empêche de continuer de jouer.

joit s'auto-appeler après x seconde. Lorsque son tour est arrivé, on parcourt chaque case du plateau et à l'aide des fonctions de test déjà écrites on vérifie pour les quatre mouvements possibles (Haut, Bas, Gauche, Droite) si un coup est possible. Si c'est le cas on le rajoute à la liste des coups possibles sous la forme [x,y,fonction] sinon on continue. Ensuite grâce à la fonction `choice` du module `random` on choisit aléatoirement dans la liste des coups possibles un coup. Ici apparaît un petit problème qui est que la fonction est de type chaîne de caractère on fait donc appel à la fonction `eval(arg1)` qui exécute la fonction qui porte le même nom que la chaîne de caractère passée en argument si elle existe sinon elle renvoie une erreur.

3.2.3 Un enregistrement, affichage des scores réalisés.

Enfin la dernière chose que j'ai voulu faire, c'est d'enregistrer les scores de façon permanente, c'est à dire que même si on quitte l'application on puisse retrouver les scores que l'on a bien voulu enregistrer. Pour cela, tout d'abord lorsque l'utilisateur finit une partie une fenêtre, `popup` apparaît lui indiquant qu'il a gagné et lui demandant si il veut enregistrer le score. Il lui suffit alors de cliquer sur oui ou non. Si l'utilisateur clique sur non rien ne se passera par contre si l'utilisateur clique sur oui on enregistrera les scores qu'il vient de réaliser dans un fichier texte, on écrit les couples clés/valeurs du dictionnaire des points séparés par des `/`, ainsi si l'utilisateur quitte l'application il retrouvera toujours ses scores. Pour les relire il lui est demandé à lire les scores on lui demandera si il veut bien que le second joueur est était l'IA ou non ainsi qu'un nom de joueur pour le retrouver dans le fichier. On lit alors le fichier et on ajoute à une liste chaque ligne, chaque score enregistré si il remplit les critères demandés par l'utilisateur.

Pour l'affichage on ouvre une autre fenêtre à l'aide de la méthode `toplevel` de `tkinter` cette méthode permet que si on ferme la fenêtre principale celle-ci se ferme. Puis à l'aide de la méthode de placement `grid` de `tkinter` on se construit un tableau (exemple ci-dessous)



<i>Nom J1</i>	<i>Nombre de boules rouges</i>	<i>Nombre de boules adverse</i>	<i>Nom J2</i>	<i>Nombre de boules rouges</i>	<i>Nombre de boules adverse</i>
marin	7	0	marin2	0	0
marin	7	0	marin2	0	0
marin	7	0	marin2	0	0
marin	7	0	marin2	0	0
marin	7	0	marin2	0	0

FIGURE 8 –

Enfin si le joueur souhaite effacer tous les scores il dispose d'une option pour le faire. En effet on ouvre alors le fichier en mode écriture ceux-ci à pour effet d'effacer le fichier.

4 Suite voulue

4.1 Une IA plus intelligente

L'IA programmée ici ne fait que tirer au sort bêtement un coup parmi tant d'autres, or j'aurais aimé pouvoir faire en sorte que l'IA choisisse d'abord en fonction des points puis ensuite au hasard. Les esquisses de programme pour implémenter cette fonctionnalité étaient de passer la liste des coups possibles dans une fonction qui regarderait le maximum de points rapportés par un coup et qui ajouterait ces coups à une liste dans laquelle serait pioché un coup pour jouer.

4.2 Une meilleur gestion des clic de la souris

Comme amélioration voulue une meilleur gestion des clic est pour moi quelque chose de très important. En effet, prenons un exemple :

- $X1=0$
- $y1=0$
- $x2=1$
- $y2=6$

D'après les coordonnées donné plus haut notre bon sens nous pousserait à dire que, malgré un petit écart sur les x , le mouvement soit vers le bas, or pour mon programme il interpréterait ces points comme étant un coup vers la droite.

4.3 Des bruitages et/ou animation

Pour rajouter un peu de vie au jeu j'aurais aimé pouvoir rajouter des bruitages/animations. J'avais comme projet d'utiliser pygame pour écouter des sons en .wav, par contre, je n'est pour l'instant aucune piste à proposer pour ajouter des animations.

5 Erreur et test de fonction représentative du programme.

Tout ce qui sera dit sont les conclusions tirées des recherches de défaut des fonctions importantes du programme effectuées dans le fichier "TestFonction.py".

5.1 anti_deplacement_contraire

La fonction est mise à mal tout d'abord lorsque l'on ne spécifie pas de "mode", en effet il n'y a aucun mode par défaut. Ensuite la plus grosse erreur produite, est que si l'on spécifie en mode écriture ("w") une liste qui n'a pas 7 éléments mais moins. Or dans la fonction anti_deplacement_contraire on regarde au rang fixe 6 et on ne s'adapte pas à la longueur de la liste.

5.2 B

cette sous-section regroupe les fonctions H, B, G, D. J'ai pu sur cette fonction trouver qu'une seule erreur. En effet celle-ci ne supporte pas que la liste, qui représente normalement la ligne ou colonne que l'on veut déplacer dans la grille, ne corresponde pas à la ligne ou colonne.

5.3 test_fin_partie

Aucune véritable erreur n'a été trouvée, seulement des incohérences. En effet il n'y a aucun test sur la cohérence des arguments passés. Ainsi les deux joueurs peuvent avoir le même nom voir un nom vide. Et, enfin, les noms qui doivent se retrouver dans les deux arguments donnés à la fonction peuvent être différents alors qu'ils ne doivent pas pouvoir l'être.

5.4 test_Pions_pousser

On retrouve dans cette fonction la même erreur que dans la fonction test_fin_partie. Mais on peut rajouter le fait si le plateau n'est pas codé correctement (ex : on change la case majuscule minuscule) les valeurs retournées deviennent complètement incohérentes.

5.5 traitement_coordone

Une unique erreur est formulée lorsque l'on spécifie un rang trop grand.

5.6 test_deplacement

J'ai pu trouver une erreur, En effet on parcourt la liste via une boucle `for ... in range()` or le range spécifié ne dépend pas de la liste passée en argument. Donc on obtient une erreur si on passe en argument une grille de 6*6 et non 7*7

5.7 `compte_pions_coul`

Comme dans la fonction `test_deplacement` le parcours du plateau via une boucle `for ... in range()` pose problème. On obtient donc une erreur si on passe en argument une grille de 6*6 et non 7*7

5.8 `search_by_name`

Ici je n'est pas réussi à trouver d'erreur cependant il y'a des incohérences à comprendre en effet on doit passer en second argument un booléen `True` ou `False`. Or on peut voir que une chaîne de caractère non vide peut remplacer le booléen `True` et au contraire une chaîne de caractère vide peut remplacer le booléen `False`.

5.9 `Grille_Initiale`

Dans cette fonction deux erreurs ont été trouvées. Tout d'abord si on passe en argument un tableau plus petit que de taille 7*7 on obtient une erreur d'index car on ne tient pas compte de la taille du tableau dans la boucle `for`. Ensuite si le tableau est plus grand que 7*7 les lignes en plus ne seront pas modifiées.

6 conclusion

Ce fut un projet fort intéressant. En effet malgré une apparence simpliste il s'est avéré difficile de construire certains points du jeu comme par exemple les fonctions de déplacement. Je suis néanmoins satisfait du résultat malgré des maladresses dans la façon de coder sur certains points. J'aurai aimé aller plus loin sur ce projet qui m'a véritablement pris à cœur de mener à terme.