




# ***B*** ***WLING*** ***Game***



Monsieur DELAY  
Monsieur DESFORETS  
Professeurs de  
Mathématique

PIGNOT Rémi TS4  
GEAY Mélanie TS4  
PAQUET Théo TS4



Pour le choix du projet, on a réfléchi à un projet original qui nous plaisait sur la réalisation.

Nous sommes très vite partis sur un jeu, nous avons tout d'abord pensé à un jeu de cartes mais ce type de projet avait déjà été utilisé par d'autre groupe les années précédentes.

Notre BrainStorming, nous a menés sur le jeu de Bowling, puis nous avons demandé aux professeurs si cela avait déjà été faits et heureusement non.

Lors de la mise en place du projet nous nous étions dit que pouvoir faire des parties à deux serait très intéressant, mais nous nous sommes vite rendus compte que cela serait trop compliqués pour nous qui étions néophyte en codage.

Faire un jeu plaisant visuellement, était un objectif secondaire. Dans ma tête je voyais un jeu très coloré comme les jeux Flash que l'on trouve sur certains sites Internet, mais la réalisation a été plus compliqué que prévu.

Même la mise en place d'une partie en 10 tours a été très compliqués car je n'étais pas très à l'aise avec le Python.




## Notre Cahier des Charges :

- Pouvoir lancer une boule sur une piste vers 10 quilles
- Pouvoir gérer à la fois l'angle et la puissance du lancer
- Pouvoir marquer de points en fonction du nombre de quilles dégommées
- Pouvoir faire des parties en Multijoueur

Notre but c'est d'avoir une impression positive et de faire réellement penser, tout du moins dans le style du jeu, à une vraie partie de Bowling.

Notre but a été au moins de pouvoir lancer la boule en pouvant dégommer les quilles.

Nous ne nous sommes pas appuyé sur quelconques jeux existants.



Je me suis concentré sur la modélisation de la piste, les quilles, de la boule, ainsi que le mouvement de la boule et les collisions entre la boule et les quilles.

Lors du codage des collisions, j'ai travaillé avec Mélanie pour le comptage des points, car Mélanie s'est occupé du tableau des scores .

Le programme s'est divisé en plusieurs parties toutes la partie Mouvement et Collisions, la partie Score et Tableau, et la partie Menu et Présentation.


La partie Menu se lance , techniquement, au lancement du programme, ensuite les parties Mouvement, Score, Tableaux se lancent simultanément et fonctionnent en collaboration.

Comme nous n'avons pas réussi à relancer une partie, le programme s'arrête à la fin du tour 10.




Afin de créer notre jeu nous avons programmé en Python sur des outils de programmation tel que Spyder, Tony EduPython.

Nous nous sommes énormément servis de tkinter qui nous a permis de faire notre interface graphique.



```
1 import tkinter as tk
2
3 Pos_Xba=720
4 Pos_Yba=130
5
6 fen = tk.Tk()
7
8 canvas = tk.Canvas(fen,width = 800, height = 310, bd=0, bg="white")
9 canvas.grid(row=1,column=1,padx=10,pady=5)
10
11 list_coord = [(20, 100), (20, 130), (20,160), (20,190), (50,115), (50,145),
12              (50,175), (80,130), (80,160), (110,145)]
13
14 lst_quilles=[canvas.create_oval(x1,y1,x1+20,y1+20,fill='black')
15              for x1, y1 in list_coord]
```


*Ceci est la création de l'interface graphique, avec la création du canvas qui sert de piste de Bowling, une liste de coordonnées qui sert de base à la liste de création des quilles avec une boucle qui crée les 10 quilles. Les 2 variables au dessus sont les coordonnées de base de la boule.*



```
28 def Lancer(event):
29     global dx, dy
30     dy = angle.get()
31     dx = -vitesse.get()
32
```

Cette fonction est particulière car elle ne s'active que lorsque le joueur appuie sur la flèche du haut de son clavier et sert à lancer la balle.

La vitesse est gérée sur l'axe x grâce à la variable "vitesse" définie plus bas et récupérée par la commande `.get`, tandis que l'angle est géré sur l'axe y grâce à la variable "angle".




```
16 def deplacement():  
• 17     global dx, dy  
• 18     X1, Y1, X2, Y2 = canvas.coords(boule)  
• 19     canvas.move(boule,dx,dy)  
• 20     fen.after(20,deplacement)  
• 21     if len(canvas.find_overlapping(X1,Y1,X2,Y2))>1:  
• 22         dy+=1  
• 23     if Y1<=0:  
• 24         dy=-angle.get()  
• 25     if Y2>=310:  
• 26         dy=angle.get()  
27
```

La fonction ci-dessus est la fonction principale du mouvement de la boule. La fonction “déplacement” sert à faire avancer la boule avec la commande “canvas.move”, mais aussi gérer certaines collisions soit avec n’importe quel objet avec la première ligne “if” qui grâce à la fonction find\_overlapping, qui détecte le recouvrement de l’objet “boule”, lorsque la boule rentre en contact avec un objet sa direction change.

Les deux autres lignes avec “if” servent à gérer les collisions avec les bords de la piste, la boule rebondit sur la paroi avec un angle de 90°.

Ici, les variables “dx” et “dy”, sont les déplacements sur l’axe X et l’axe Y de la boule.





```

47
• 48 tr1 = tk.IntVar(0)
• 49 tr2 = tk.IntVar(0)
• 50 tr3 = tk.IntVar(0)
• 51 tr4 = tk.IntVar(0)
• 52 tr5 = tk.IntVar(0)
• 53 tr6 = tk.IntVar(0)
• 54 tr7 = tk.IntVar(0)
• 55 tr8 = tk.IntVar(0)
• 56 tr9 = tk.IntVar(0)
• 57 tr10 = tk.IntVar(0)
58
• 59 tr1.set(0)
• 60 tr2.set(0)
• 61 tr3.set(0)
• 62 tr4.set(0)
• 63 tr5.set(0)
• 64 tr6.set(0)
• 65 tr7.set(0)
• 66 tr8.set(0)
• 67 tr9.set(0)
• 68 tr10.set(0)
69

```

La création des variables nécessaires à la création du tableau. Ces variables seront changés afin de mettre à jour les scores dans le tableau, de la façon ci dessous.

```

if len(canvas.find_overlapping(110
    canvas.delete(lst_quilles[9])
    tr1.set(tr1.get()+1)

```

Ici la commande .set permet de d'associer la variable à un nombre tandis que la commande .get permet de récupérer le chiffre associé.

```

69
70 #Score total
• 71 total = tr1.get()+tr2.get()+tr3.get()+tr4.get()+tr5.get()+tr6.get()+tr7.get()+tr8.get()+tr9.get()+tr10.get()
72
73 #Création du tableau
• 74 tk.Label(fen, text='Tour', borderwidth=1).grid(row=3, column=8)
• 75 tk.Label(fen, text='J1', borderwidth=1).grid(row=4, column=8)
76
• 77 tk.Label(fen, text=total, borderwidth=3).grid(row=4, column=20)
• 78 tk.Label(fen, text='Total', borderwidth=3).grid(row=3, column=20)
79
• 80 for colonne in range(10, 20):
• 81     tk.Label(fen, text=str(colonne-9), borderwidth=3).grid(row=3, column=colonne)
• 82     tk.Label(fen, textvar=tr1, borderwidth=3).grid(row=4, column=10)
• 83     tk.Label(fen, textvar=tr2, borderwidth=3).grid(row=4, column=11)
• 84     tk.Label(fen, textvar=tr3, borderwidth=3).grid(row=4, column=12)
• 85     tk.Label(fen, textvar=tr4, borderwidth=3).grid(row=4, column=13)
• 86     tk.Label(fen, textvar=tr5, borderwidth=3).grid(row=4, column=14)
• 87     tk.Label(fen, textvar=tr6, borderwidth=3).grid(row=4, column=15)
• 88     tk.Label(fen, textvar=tr7, borderwidth=3).grid(row=4, column=16)
• 89     tk.Label(fen, textvar=tr8, borderwidth=3).grid(row=4, column=17)
• 90     tk.Label(fen, textvar=tr9, borderwidth=3).grid(row=4, column=18)
• 91     tk.Label(fen, textvar=tr10, borderwidth=3).grid(row=4, column=19)
92
93

```

La création du tableau en lui même prenant en compte les variables créée plus haut, le tableau est organisé en ligne et en colonne grâce au .grid.

Mélanie a utilisé une boucle afin de créer la ligne supérieur du tableau.

La variable total est le score final au bout des 10 tours.

La commande .get est utilisé afin de récupérer le chiffre correspondant au tour indiqué.

```

98
• 99 angle = tk.DoubleVar(value=0)
• 100 vitesse= tk.DoubleVar(value=0)
• 101 scaledir=tk.Scale(fen, orient='vertical', from_=10, to=-10,
• 102     resolution=1, length=250,
• 103     variable = angle,
• 104     label='Direction',)
• 105 scaledir.grid(row=1,column=6,padx=15)
106
• 107 scalePui=tk.Scale(fen, orient='horizontal', from_=10, to=1,
• 108     resolution=1, length=250,
• 109     variable = vitesse,
• 110     label='Puissance')
• 111 scalePui.grid(row=2,column=1,pady=5)
112
113

```

Ce morceau de code est la création, des 2 Scales qui permettent de gérer de façon intuitive pour l'un la puissance et pour l'autre la direction.

Pour cela j'ai utilisé la fonction scale de Tkinter.

J'ai récupérer ces valeurs des boutons Scale dans des variables, que j'ai utilisé dans différentes fonctions.

```

• 20     canvas.move(boat, dx, dy)
• 21     fen.after(20,deplacement)
• 22     if len(canvas.find_overlapping(X1,Y1,X2,Y2))>1:
• 23         dy=+1
• 24     if Y1<=0:
• 25         dy=-angle.get()
• 26     if Y2>=10:
• 27         dy=angle.get()
28
29 def Lancer(event):
30     global dx, dy
31     dy = angle.get()
32     dx = -vitesse.get()

```

```

116
117
• 118 Bouton_Quitter=tk.Button(fen, text ='Quitter', command = fen.destroy)
• 119 Bouton_Quitter.grid(row=5,column=1,sticky='')
120
• 121 canvas.bind_all('<Up>', Lancer)
122
• 123 tk.Button(fen,text='Bas',command=Bas).grid(row=1,column=2,sticky='sw',
• 124         padx=5,pady=5)
• 125 tk.Button(fen,text='Haut',command=Haut).grid(row=1,column=2,sticky='nw',
• 126         padx=5,pady=5)
127

```

Ici des boutons gérant différentes fonctions sont créés à l'aide de la commande Button de tkinter.

Le premier sert à quitter la fenêtre et le programme à l'aide de la commande "fen.destroy".

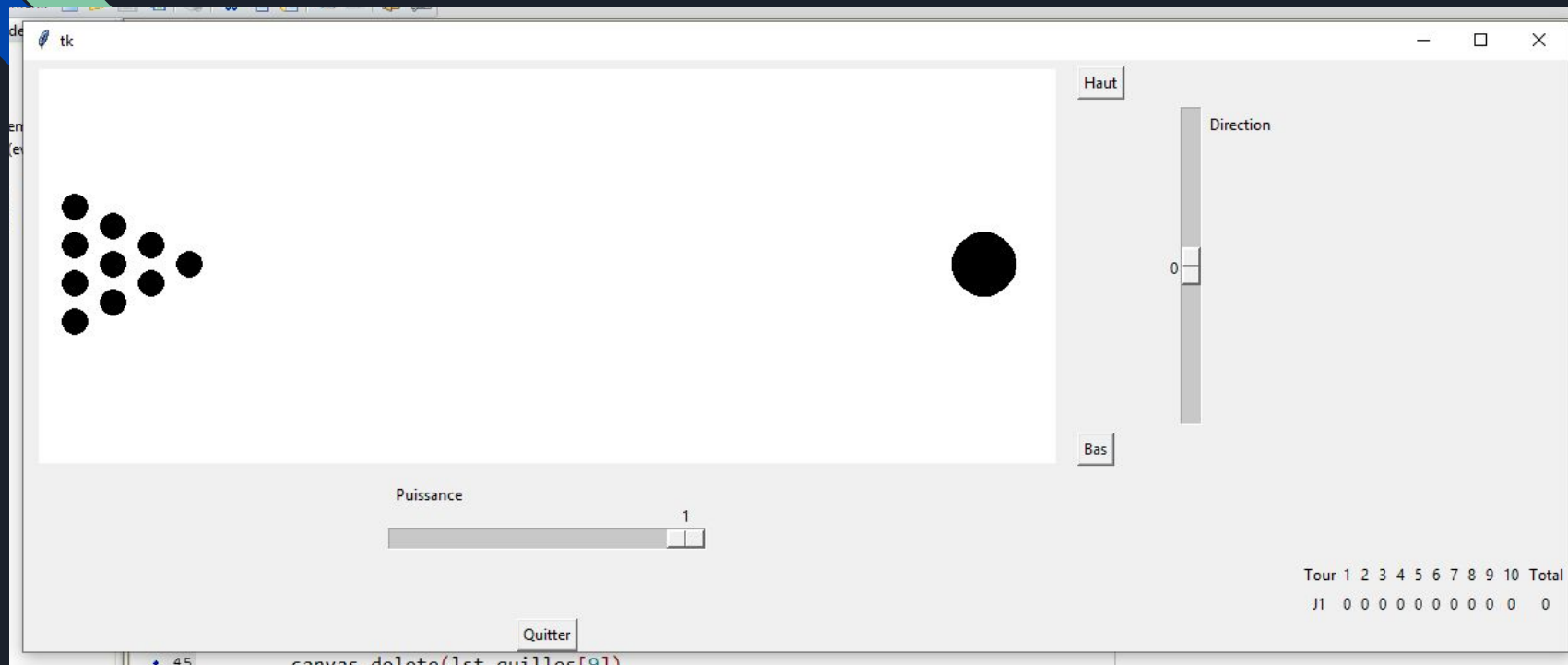
La commande ligne 121 sert à lancer la fonction "Lancer" lorsque le joueur appuie sur la flèche du haut sur son clavier grâce à la commande bind\_all.

Les deux autres boutons sont utilisables seulement sur l'interface tkinter, et sont utilisés pour déplacer la boule afin de la positionner grâce au fonction "Haut" et "Bas" qui influent directement sur les coordonnées de la boule.

```

32
33 def Bas():
• 34     X1, Y1, X2, Y2 = canvas.coords(boule)
• 35     canvas.coords(boule,X1,Y1+20,X2,Y2+20)
36
37 def Haut():
• 38     X1, Y1, X2, Y2 = canvas.coords(boule)
• 39     canvas.coords(boule,X1,Y1-20,X2,Y2-20)
40

```



• 45

`canvas.delete(1st_guilles[91])`



Nos plus grandes difficultés sont :

- Le fait de réussir à créer une partie de 10 tours
- Marquer les points lors d'une collision
- La création d'un Menu fonctionnelle

Je travaille toujours sur ces problèmes et j'espère pouvoir les résoudre avant la soutenance.



Nous nous sommes tous retrouvé etentraide afin de mettre en commun nos différentes parties de code, Discord nous a beaucoup aidé à communiquer à distance à l'aide de conversations en direct, a vrai dire nous nous sommes bien plus servis de Discord pour partagé nos fichiers que GitHub.

J'ai été le plus actif dans la mise en commun car j'étais le plus à l'aise sur Python.



Le programme à l'heure actuelle ne respecte qu'en partie le cahier des charges, et il y a beaucoup de chose à améliorer et à compléter.

Mon objectif d'ici Jeudi c'est de réussir à faire une partie de 10 tours et d'intégrer le Menu.

Les autres points à améliorer sont la beauté du jeu et les collisions parfois pas très réalistes.





**Le projet m'as permis de m'exercer et d'apprécier davantage le codage.**

**Il m'as aussi montré la difficulté de coder et de tenir des délais lorsque cela concerne de gros projets.**

**Il m'as aussi permet de voir comment bien s'organiser et bien communiquer afin de bien travailler en équipe.**



Webographie:

<https://python-django.dev/page-tkinter-interface-graphique-python-tutoriel> pour toutes la partie tkinter.

<http://tableauxmaths.fr/spip/spip.php?article48> pour tout ce qui est collisions.