

# *Jeu de bowling*



Groupe n°1 : Rémi PIGNOT – Théo PAQUET – Mélanie GEAY  
(Terminale S4)

Professeurs : M. DELAY et M. DESFORÊTS

## *Sommaire*

Présentation du projet	Page 3
Répartition des tâches	Page 4
Programmation	Page 5
Mise en commun	Page 10
Programme final	Page 11
Bilan du projet	Page 12
Annexes	Page 13

# *Présentation du projet*

## *Choix du projet*

Pour le choix du projet, nous avons décidé de faire un jeu, ce qui nous paraissait plaisant à réaliser. Nous avons d'abord pensé à un jeu de cartes, mais cela étant peu original, nous nous sommes tournés vers un jeu de bowling, ce qui sortait plus de l'ordinaire.

## *Problématique*

Le but de ce projet est de créer un jeu de bowling sous Python muni d'une interface Tkinter afin de pouvoir jouer à ce jeu même lorsque l'on souhaite ne pas se déplacer ou payer pour jouer au bowling.

## *Cahier des charges*

Afin de réaliser notre jeu de bowling, nous allons d'abord créer une piste contenant 10 quilles et une boule. La boule doit pouvoir se déplacer jusqu'aux quilles qui devront disparaître lorsqu'elles auront été touchées.

Afin de lancer la boule, il y aura deux jauges : une pour la puissance et une autre pour décider de l'angle de notre lancer. De plus, il sera possible de se déplacer avec la boule sur la gauche ou la droite de la piste.

Le jeu se fera en 10 manches avec 2 lancers par manche (sauf

strike où dans ce cas il y aura 1 lancer). Seulement la dernière (10e manche) pourra se faire en 3 lancers en faisant un strike ou un spare au 1er ou 2e lancer.

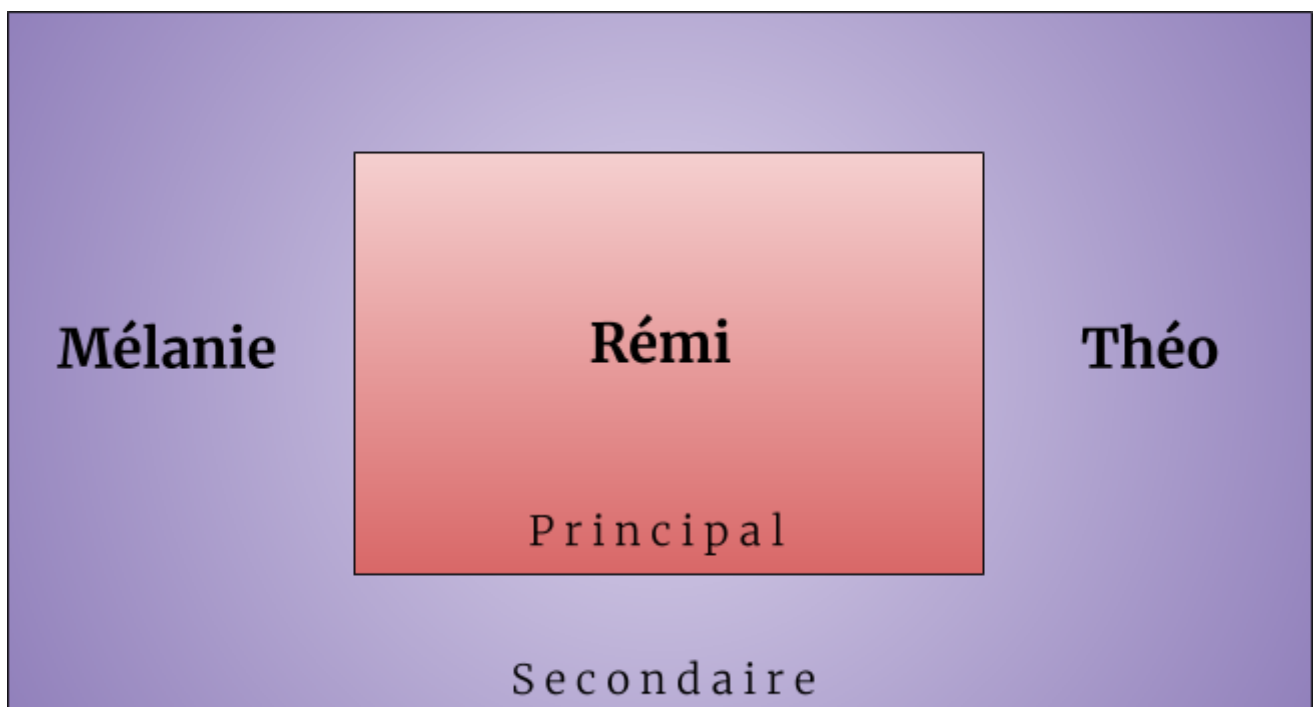
Il faudra donc créer un tableau pour recenser les scores obtenus à chaque lancer et pour afficher le score final de la partie.

## Répartition des tâches

Afin d'oeuvrer à la bonne réalisation de notre projet, nous avons décidé de répartir les tâches du jeu de la manière suivante :

- **Rémi** s'occupe du programme principal, c'est-à-dire de créer la piste, la boule, les quilles et les effets associés à la boule et aux quilles.
- **Théo** s'occupe de l'un des effets donnés à la boule : la jauge de puissance.
- **Mélanie** se charge de la création du menu principal, du calcul des points à chaque manche/lancer et de l'affichage du tableau de scores recensant les points gagnés à chaque manche et le score final.

Autrement dit, la répartition des tâches se résume à ce schéma :



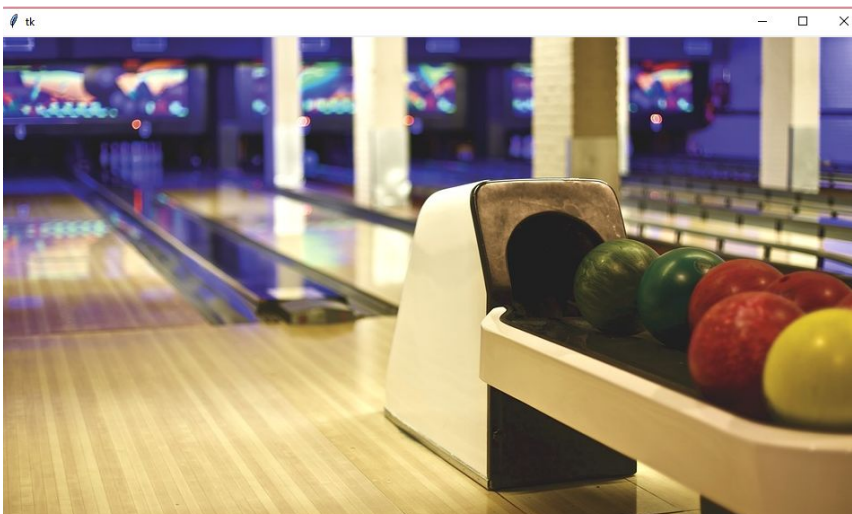
## *Programmation*

Afin de programmer, j'ai utilisé Python en important une interface Tkinter sur différents éditeurs (EduPython, Thonny, Spyder).

### *Menu principal*

J'ai d'abord procédé à l'affichage de l'image, ce qui a été compliqué à réaliser. En effet, tous les formats d'images ne sont pas acceptés, j'ai donc dû en tester plusieurs. De plus, il fallait dimensionner parfaitement l'image par rapport à la fenêtre, sinon elle disparaissait. Un autre problème étant l'affichage de messages d'erreur sur certains éditeurs alors que sur d'autres le programme fonctionne parfaitement.

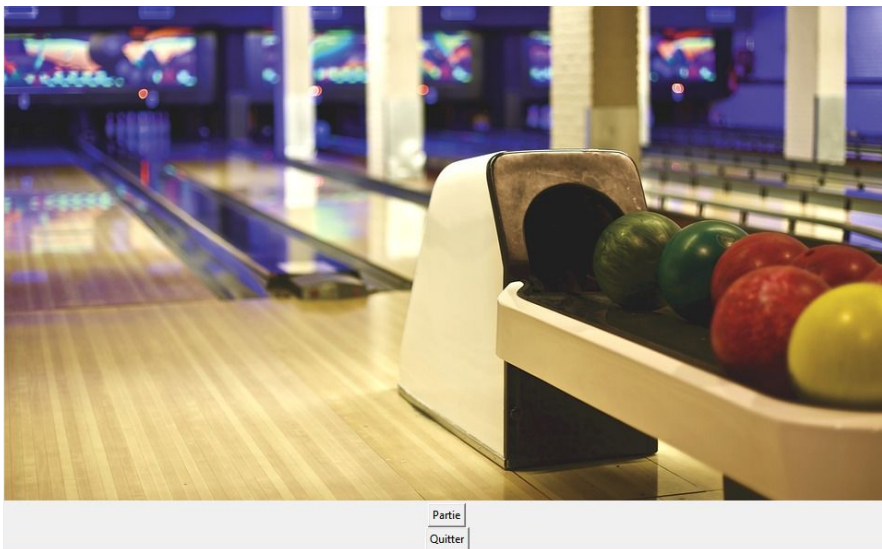
```
photo = tk.PhotoImage(file="B.png")  
  
canvas = tk.Canvas(fenetre, width=960, height=539)  
canvas.create_image(480, 270, image=photo)  
canvas.pack()
```



La création de boutons dans le menu pour jouer ou quitter la partie a été plus rapide. Les programmes n'étant pas encore mis en commun avec les autres membres du groupe, le bouton "partie" n'est pas encore relié à la fonction pour lancer la partie de bowling.

```
Bouton_Partie = tk.Button(fenetre, text = 'Partie', command = partie)
Bouton_Partie.pack()

Bouton_Quitter = tk.Button(fenetre, text = 'Quitter', command = fenetre.destroy)
Bouton_Quitter.pack()
```



### Tableau des scores

Afin de rassembler les points gagnés dans la partie, j'ai procédé à la création du tableau des scores.

J'ai d'abord créé 10 variables, une pour chaque manche, que j'initialise à 0.

```
#Scores manches 1 à 10
tr1 = tk.IntVar(0)
tr2 = tk.IntVar(0)
tr3 = tk.IntVar(0)
tr4 = tk.IntVar(0)
tr5 = tk.IntVar(0)
tr6 = tk.IntVar(0)
tr7 = tk.IntVar(0)
tr8 = tk.IntVar(0)
tr9 = tk.IntVar(0)
tr10 = tk.IntVar(0)
```

Ensuite, il a fallu retrouver le nombre de quilles que la boule avait fait tomber pour pouvoir le mettre dans le futur tableau des scores. Puisque toutes les parties du programme final n'ont pas été mis en commun, j'ai pour l'instant mis des valeurs au hasard pour chaque tour qui à la fin seront remplacées par les noms des variables contenant le nombre de quilles tombées.

```
tr1.set(2)
tr2.set(8)
tr3.set(6)
tr4.set(3)
tr5.set(9)
tr6.set(2)
tr7.set(0)
tr8.set(10)
tr9.set(5)
tr10.set(1)
```

J'ai ensuite créé une variable afin d'additionner les valeurs précédentes pour calculer le score total.

```
#Score total
total = tr1.get()+tr2.get()+tr3.get()+tr4.get()+tr5.get()+tr6.get()+tr7.get()+tr8.get()+tr9.get()+tr10.get()
```



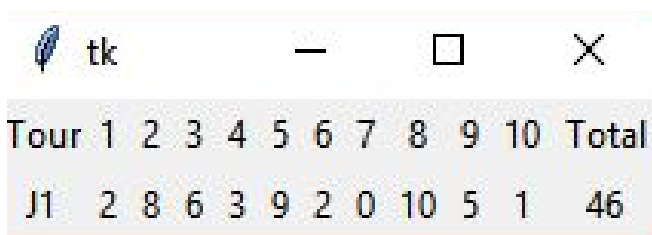
Puis j'ai procédé à la création des premiers éléments du tableau, c'est à dire les cases « Tour », « Joueur » et « Total ».

```
#Création du tableau
tk.Label(fenetre, text='Tour', borderwidth=1).grid(row=1, column=1)
tk.Label(fenetre, text='J1', borderwidth=1).grid(row=2, column=1)

tk.Label(fenetre, text=total, borderwidth=3).grid(row=2, column=12)
tk.Label(fenetre, text='Total', borderwidth=3).grid(row=1, column=12)
```

Après, j'ai fait en sorte d'afficher les valeurs entrées précédemment dans le tableau puis de les mettre sous le numéro de chaque tour correspondant grâce à une boucle.

```
for colonne in range(2, 12):
    tk.Label(fenetre, text=str(colonne-1), borderwidth=3).grid(row=1, column=colonne)
    tk.Label(fenetre, textvar=tr1, borderwidth=3).grid(row=2, column=2)
    tk.Label(fenetre, textvar=tr2, borderwidth=3).grid(row=2, column=3)
    tk.Label(fenetre, textvar=tr3, borderwidth=3).grid(row=2, column=4)
    tk.Label(fenetre, textvar=tr4, borderwidth=3).grid(row=2, column=5)
    tk.Label(fenetre, textvar=tr5, borderwidth=3).grid(row=2, column=6)
    tk.Label(fenetre, textvar=tr6, borderwidth=3).grid(row=2, column=7)
    tk.Label(fenetre, textvar=tr7, borderwidth=3).grid(row=2, column=8)
    tk.Label(fenetre, textvar=tr8, borderwidth=3).grid(row=2, column=9)
    tk.Label(fenetre, textvar=tr9, borderwidth=3).grid(row=2, column=10)
    tk.Label(fenetre, textvar=tr10, borderwidth=3).grid(row=2, column=11)
```



Tour	1	2	3	4	5	6	7	8	9	10	Total
J1	2	8	6	3	9	2	0	10	5	1	46

## *Mise en commun*

Afin de mettre en commun nos différentes parties de programme, nous avons communiqué principalement à l'aide de Discord. Rémi a été le plus actif dans la mise en commun car il est le plus à l'aise avec Python.

- Concernant le **menu principal**, le bouton “Partie” n’a pas été relié au programme de jeu, cependant nous allons travailler pour résoudre ce problème et le lier à la fonction “partie” qui lancera le programme principal .
- Concernant le **tableau des scores**, il s’est révélé plus difficile de le lier au score réel car Rémi a rencontré des difficultés pour ce qui est du marquage des points lors d’une collision, ce qu’il espère pouvoir résoudre prochainement.

## Programme final

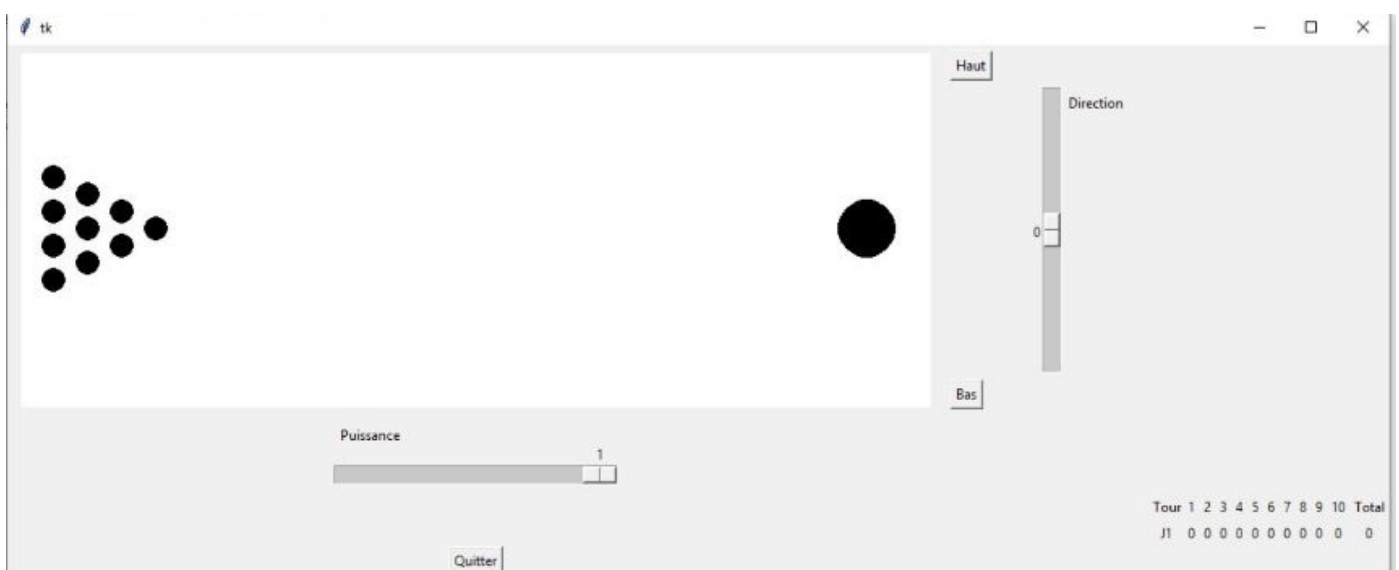
Notre programme ne répond pas à tous les critères du cahier des charges.

En effet, après nos derniers tests, nous avons constaté que le jeu fonctionnait :

- le jeu contient une piste avec 10 quilles et une boule
- la boule peut être dirigée comme l'on souhaite et lancée avec une certaine puissance
- il y a un tableau de scores pouvant calculer le score total

Cependant, certains éléments ne fonctionnent pas comme nous l'aurions souhaité :

- le calcul des scores lors d'une collision n'est pas encore au point
- le tableau des scores ne peut donc pas encore afficher les scores obtenus sur les 10 tours



## *Bilan du projet*

### *Bilan du projet, possibles améliorations :*

Le jeu de bowling fonctionne malgré des éléments manquants, cependant nous aurions pu rajouter des détails qui auraient rendu le jeu plus plaisant : l'interface de jeu n'est pas très jolie, le calcul des points ne prend pas en compte toutes les règles (après un strike ou un spare) et il n'est possible de jouer qu'en solo.

### *Bilan personnel*

Ce projet m'a permis de me rendre compte que programmer un jeu n'était pas quelque chose de facile, qu'il faut savoir être organisé au sein de son équipe, persévérant et critique sur son projet afin de pouvoir s'améliorer et corriger ses erreurs qui ne sont pas toujours faciles à trouver. J'ai pu remarquer qu'un jeu, bien qu'il paraisse simple et peu joli, demande beaucoup de programmation, de temps. Je me suis donc rendue qu'il n'était pas possible pour mon niveau de créer un jeu de bowling digne des jeux flash, bien que cela m'ait fait progresser en programmation.

# Annexes

```
import
tkinter as
tk

Pos_X=720 #Position Initial de la boule sur l'axe X
Pos_Y=130 #Position Initial de la boule sur l'axe Y

fen = tk.Tk()

#Création du Canvas qui sert de Piste
canvas = tk.Canvas(fen,width = 800, height = 310, bd=0, bg="white")
canvas.grid(row=1,column=1,padx=10,pady=5)

#Création d'une liste de coordonnées Initiale des quilles
list_coord = [(20, 100), (20, 130), (20,160), (20,190), (50,115),
(50,145),
                (50,175), (80,130), (80,160), (110,145)]

#Création des quilles
lst_quilles=[canvas.create_oval(x1,y1,x1+20,y1+20,fill='black')
              for x1, y1 in list_coord]

def deplacement():
    """Gère les déplacements de la boule et ses collisions avec les
    objets et
    les bords de la piste"""
    global dx, dy
    X1, Y1, X2, Y2 = canvas.coords(boule)
    canvas.move(boule,dx,dy)
    fen.after(20,deplacement)

    #Change la direction de laboule si elle rentre en contact avec un
    objet
    if len(canvas.find_overlapping(X1,Y1,X2,Y2))>1:

        dy=+1

    # Fais rebondir la boule sur les bords de la piste en fonction de
    son angle d'arrivée
```

```

    if Y1<=0:
        dy=-angle.get()

    if Y2>=310:
        dy=angle.get()

def Lancer(event):
    """ Permet de lancer la Boule en appuyant sur la flèche du haut"""
    global dx, dy
    dy = angle.get()
    dx = -vitesse.get()

def Bas():
    """Permet de déplacer la boule vers le Bas"""
    X1, Y1, X2, Y2 = canvas.coords(boule)
    canvas.coords(boule,X1,Y1+20,X2,Y2+20)

def Haut():
    """Permet de déplacer la boule vers le Haut"""
    X1, Y1, X2, Y2 = canvas.coords(boule)
    canvas.coords(boule,X1,Y1-20,X2,Y2-20)

def score():
    """Permet de marquer des points en fonctions des collisions des
    quilles"""
    X1, Y1, X2, Y2 = canvas.coords(boule)

    if len(canvas.find_overlapping(110,145,130,165))>1: # Détecte les
    collisions avec la quille
        canvas.delete(lst_quilles[9]) # Supprime la quille touchée

        tr1.set(tr1.get()+1) # Ajoute plus 1 au score si la quille est
        touchée
    # Création des variables scores
    tr1 = tk.IntVar(0)

```

```
tr2 = tk.IntVar(0)

tr3 = tk.IntVar(0)

tr4 = tk.IntVar(0)

tr5 = tk.IntVar(0)

tr6 = tk.IntVar(0)

tr7 = tk.IntVar(0)

tr8 = tk.IntVar(0)

tr9 = tk.IntVar(0)

tr10 = tk.IntVar(0)
```

```
tr1.set(0)

tr2.set(0)

tr3.set(0)

tr4.set(0)

tr5.set(0)

tr6.set(0)

tr7.set(0)

tr8.set(0)

tr9.set(0)

tr10.set(0)
```

```
#Score total
```

```
total =

tr1.get()+tr2.get()+tr3.get()+tr4.get()+tr5.get()+tr6.get()+tr7.get()+
tr8.get()+tr9.get()+tr10.get()
```

```
#Création du tableau
```

```
tk.Label(fen, text='Tour', borderwidth=1).grid(row=3, column=8)

tk.Label(fen, text='J1', borderwidth=1).grid(row=4, column=8)
```

```
tk.Label(fen, text=total, borderwidth=3).grid(row=4, column=20)

tk.Label(fen, text='Total', borderwidth=3).grid(row=3, column=20)
```

```

for colonne in range(10, 20):

    tk.Label(fen, text=str(colonne-9), borderwidth=3).grid(row=3,
column=colonne)
    tk.Label(fen, textvar=tr1, borderwidth=3).grid(row=4, column=10)
    tk.Label(fen, textvar=tr2, borderwidth=3).grid(row=4, column=11)
    tk.Label(fen, textvar=tr3, borderwidth=3).grid(row=4, column=12)
    tk.Label(fen, textvar=tr4, borderwidth=3).grid(row=4, column=13)
    tk.Label(fen, textvar=tr5, borderwidth=3).grid(row=4, column=14)
    tk.Label(fen, textvar=tr6, borderwidth=3).grid(row=4, column=15)
    tk.Label(fen, textvar=tr7, borderwidth=3).grid(row=4, column=16)
    tk.Label(fen, textvar=tr8, borderwidth=3).grid(row=4, column=17)
    tk.Label(fen, textvar=tr9, borderwidth=3).grid(row=4, column=18)
    tk.Label(fen, textvar=tr10, borderwidth=3).grid(row=4, column=19)


# Création de la Boule

boule = canvas.create_oval(Pos_X,Pos_Y,Pos_X+50,Pos_Y+50,
fill='black')


angle = tk.DoubleVar(value=0)# Variable qui représente l'angle de la
boule
vitesse= tk.DoubleVar(value=0)# Variable qui représente la vitesse de
la boule
#Création des Boutons Scales permettant de gérer la direction et la
vitesse de la boule
scaledir=tk.Scale(fen, orient='vertical', from_=10, to=-10,
resolution=1, length=250,
variable = angle,
label='Direction',)
scaledir.grid(row=1,column=6,padx=15)


scalePui=tk.Scale(fen, orient='horizontal', from_=10, to=1,
resolution=1, length=250,
variable = vitesse,
label='Puissance')

```



```
scalePui.grid(row=2,column=1,pady=5)
```

```
dx=0
```

```
dy=0
```

```
# Création du Bouton permettant de fermer le programme
```

```
Bouton_Quitter=tk.Button(fen, text ='Quitter', command = fen.destroy)
```

```
Bouton_Quitter.grid(row=5,column=1,sticky='')
```

```
#Permet de récupérer le fait que le joueur appuie sur le bouton  
"fleche du haut"
```

```
canvas.bind_all('<Up>', Lancer)
```

```
# Création des Boutons qui permettent de lancer les fonctions "Haut"  
et "Bas"
```

```
tk.Button(fen,text='Bas',command=Bas).grid(row=1,column=2,sticky='sw',  
padx=5,pady=5)
```

```
tk.Button(fen,text='Haut',command=Haut).grid(row=1,column=2,sticky='nw',  
,  
padx=5,pady=5)
```

```
deplacement()
```

```
score()
```

```
fen.mainloop()
```

```
import  
tkinter as  
tk
```

```
Pos_X=720 #Position Initial de la boule sur l'axe X
```

```
Pos_Y=130 #Position Initial de la boule sur l'axe Y
```

```

fen = tk.Tk()

#Création du Canvas qui sert de Piste
canvas = tk.Canvas(fen,width = 800, height = 310, bd=0, bg="white")
canvas.grid(row=1,column=1,padx=10,pady=5)

#Création d'une liste de coordonnées Initiale des quilles
list_coord = [(20, 100), (20, 130), (20,160), (20,190), (50,115),
(50,145),
                (50,175), (80,130), (80,160), (110,145)]

#Création des quilles
lst_quilles=[canvas.create_oval(x1,y1,x1+20,y1+20,fill='black')
                for x1, y1 in list_coord]

def deplacement():
    """Gère les déplacements de la boule et ses collisions avec les
    objets et
    les bords de la piste"""
    global dx, dy
    X1, Y1, X2, Y2 = canvas.coords(boule)
    canvas.move(boule,dx,dy)
    fen.after(20,deplacement)

    #Change la direction de la boule si elle rentre en contact avec un
    objet
    if len(canvas.find_overlapping(X1,Y1,X2,Y2))>1:

        dy+=1

    # Fais rebondir la boule sur les bords de la piste en fonction de
    son angle d'arrivée
    if Y1<=0:
        dy=-angle.get()

    if Y2>=310:
        dy=angle.get()

def Lancer(event):
    """ Permet de lancer la Boule en appuyant sur la flèche du haut"""
    global dx, dy
    dy = angle.get()

```

```

dx = -vitesse.get()

def Bas():

    """Permet de déplacer la boule vers le Bas"""

    X1, Y1, X2, Y2 = canvas.coords(boule)

    canvas.coords(boule,X1,Y1+20,X2,Y2+20)

def Haut():

    """Permet de déplacer la boule vers le Haut"""

    X1, Y1, X2, Y2 = canvas.coords(boule)

    canvas.coords(boule,X1,Y1-20,X2,Y2-20)

def score():

    """Permet de marquer des points en fonctions des collisions des
    quilles"""
    X1, Y1, X2, Y2 = canvas.coords(boule)

    if len(canvas.find_overlapping(110,145,130,165))>1: # Détecte les
    collisions avec la quille
        canvas.delete(lst_quilles[9]) # Supprime la quille touchée

        tr1.set(tr1.get()+1) # Ajoute plus 1 au score si la quille est
        touchée
    # Création des variables scores

    tr1 = tk.IntVar(0)

    tr2 = tk.IntVar(0)

    tr3 = tk.IntVar(0)

    tr4 = tk.IntVar(0)

    tr5 = tk.IntVar(0)

    tr6 = tk.IntVar(0)

    tr7 = tk.IntVar(0)

    tr8 = tk.IntVar(0)

    tr9 = tk.IntVar(0)

    tr10 = tk.IntVar(0)

```

```

tr1.set(0)

tr2.set(0)

tr3.set(0)

tr4.set(0)

tr5.set(0)

tr6.set(0)

tr7.set(0)

tr8.set(0)

tr9.set(0)

tr10.set(0)


#Score total

total =

tr1.get()+tr2.get()+tr3.get()+tr4.get()+tr5.get()+tr6.get()+tr7.get()+
tr8.get()+tr9.get()+tr10.get()


#Création du tableau

tk.Label(fen, text='Tour', borderwidth=1).grid(row=3, column=8)

tk.Label(fen, text='J1', borderwidth=1).grid(row=4, column=8)


tk.Label(fen, text=total, borderwidth=3).grid(row=4, column=20)

tk.Label(fen, text='Total', borderwidth=3).grid(row=3, column=20)


for colonne in range(10, 20):

    tk.Label(fen, text=str(colonne-9), borderwidth=3).grid(row=3,
column=colonne)

    tk.Label(fen, textvar=tr1, borderwidth=3).grid(row=4, column=10)

    tk.Label(fen, textvar=tr2, borderwidth=3).grid(row=4, column=11)

    tk.Label(fen, textvar=tr3, borderwidth=3).grid(row=4, column=12)

    tk.Label(fen, textvar=tr4, borderwidth=3).grid(row=4, column=13)

    tk.Label(fen, textvar=tr5, borderwidth=3).grid(row=4, column=14)

    tk.Label(fen, textvar=tr6, borderwidth=3).grid(row=4, column=15)

    tk.Label(fen, textvar=tr7, borderwidth=3).grid(row=4, column=16)

```

```

tk.Label(fen, textvar=tr8, borderwidth=3).grid(row=4, column=17)

tk.Label(fen, textvar=tr9, borderwidth=3).grid(row=4, column=18)

tk.Label(fen, textvar=tr10, borderwidth=3).grid(row=4, column=19)


# Création de la Boule

boule = canvas.create_oval(Pos_X,Pos_Y,Pos_X+50,Pos_Y+50,
fill='black')


angle = tk.DoubleVar(value=0)# Variable qui représente l'angle de la
boule
vitesse= tk.DoubleVar(value=0)# Variable qui représente la vitesse de
la boule
#Création des Boutons Scales permettant de gérer la direction et la
vitesse de la boule
scaledir=tk.Scale(fen, orient='vertical', from_=10, to=-10,
resolution=1, length=250,
variable = angle,
label='Direction',)
scaledir.grid(row=1,column=6,padx=15)


scalePui=tk.Scale(fen, orient='horizontal', from_=10, to=1,
resolution=1, length=250,
variable = vitesse,
label='Puissance')
scalePui.grid(row=2,column=1,pady=5)


dx=0
dy=0


# Création du Bouton permettant de fermer le programme
Bouton_Quitter=tk.Button(fen, text ='Quitter', command = fen.destroy)

```

```
Bouton_Quitter.grid(row=5,column=1,sticky='')

#Permet de récupérer le fait que le joueur appuie sur le bouton
"fleche du haut"
canvas.bind_all('<Up>', Lancer)

# Création des Boutons qui permettent de lancer les fonctions "Haut"
et "Bas"
tk.Button(fen,text='Bas',command=Bas).grid(row=1,column=2,sticky='sw',
      padx=5,pady=5)

tk.Button(fen,text='Haut',command=Haut).grid(row=1,column=2,sticky='nw
',
      padx=5,pady=5)


deplacement()

score()


fen.mainloop()
```