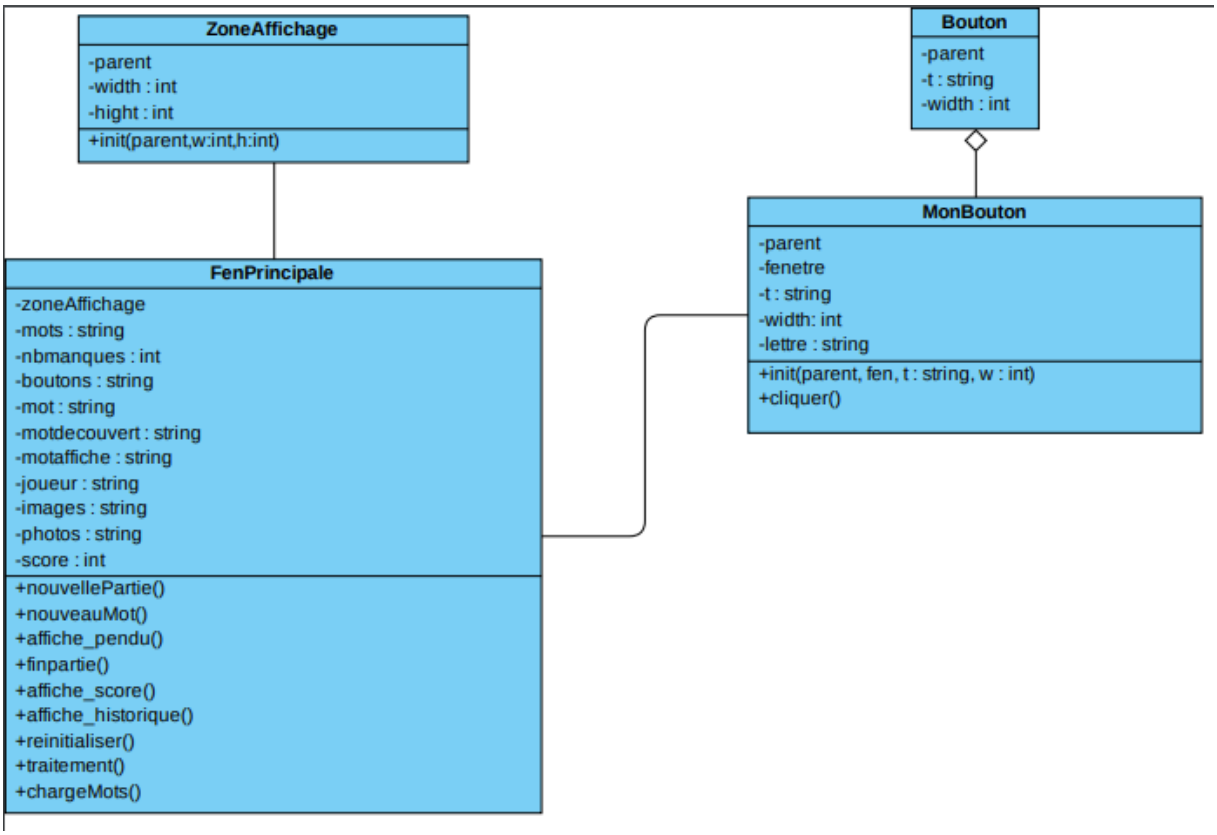
GIULIANI Mathilde
SCHIEBER Manon

Groupe de TD D2b, Mr CHALON

Rapport TD5 : Jeu du pendu

Diagramme des classes UML :



Le diagramme UML ci-dessus est celui correspondant au fichier « pendu_images.py ». Dans le cas du fichier « pendu_formes.py », la fonction affiche_pendu() de la classe FenPrincipale est remplacée par la fonction affiche_forme().

Code source commenté :

Création des boutons principaux

```
20      #On crée les boutons principaux
21      f = Frame(self)
22      f.pack(side=BOTTOM, padx=5, pady=5)
23      Button(f, text ='Nouvelle partie', width=15, command =
24      self.nouvellePartie).pack(side=LEFT, padx = 5,pady = 5)
25      Button(f, text='Quitter',width=12, command=self.destroy).pack(side=LEFT, padx=5, pady=5)   #
26      Button(f, text='Afficher score',width=12, command=self.affiche_score).pack(side=LEFT, padx=5, pady=5)
27      Button(f, text='Historique',width=12, command=self.affiche_historique).pack(side=LEFT, padx=5, pady=5)
28      Button(f, text='Réinitialiser',width=12, command=self.reinitialiser).pack(side=LEFT, padx=5, pady=5)
```

On utilise la classe Button.

1. Création des boutons du clavier

```
30      #On crée les boutons du clavier
31      lettres = []
32      for i in range(26):
33          lettres.append(chr(ord('A')+i))    #code ASCI
34
35      self.__boutons=[]
36      f1 = Frame(self)
37      f1.pack(side=TOP, padx=5, pady=5)
38      for i in range(1,27):
39          bouton = MonBouton(f1,self,lettres[i-1],4)
40          bouton.grid(row=(i//8),column= i%8)
41          self.__boutons.append(bouton)
42          bouton.config(command = bouton.cliquer)
```

On créé une liste de lettres, puis on créé chaque bouton associé à une lettre en lui assignant la commande « cliquer », définie dans la classe « MonBouton » :

```
213 class MonBouton(Button):
214     def __init__(self,parent, fen, t, w):
215         Button.__init__(self,master=parent,text=t,width=w)
216         self.__fenetrePrincipale =fen
217         self.__lettre = t
218
219
220     def cliquer(self):
221         self.config(state=DISABLED)   #pour désactiver le bouton quand on a cliqué dessus
222         self.__fenetrePrincipale.traitement(self.__lettre)
```

Chaque fois qu'on cliquera sur un bouton, le jeu va désactiver ce bouton et traiter le cas de la sélection de la lettre associée au bouton.

2. Méthode nouvelle_partie

```
75      def nouvellePartie(self):
76          self.__mot=self.nouveauMot()
77          self.__motdecouvert='*'*len(self.__mot)
78          self.__motaffiche.config(text='Mot :'+self.__motdecouvert)
79
80          for i in range(1,27):
81              self.__boutons[i-1].config(state='normal')    #on réactive tous les boutons
82
83          self.__nbmanques=0  #le nombre d'échecs retombe à 0
84          self.affiche_pendu()   #on réaffiche la première image
```

Elle détermine un nouveau mot à deviner, réactive tous les boutons du clavier, réinitialise le nombre de fautes à 0 et affiche l'image initiale du pendu (fond blanc).

3. Méthodes nouveauMot et chargeMot

```
87      def nouveauMot(self):   #choisit un mot au hasard dans la liste
88          l = len(self.__mots)
89          n = randint(0,l-1)
90          mot = self.__mots[n]
91          return mot
92
93
94      def chargeMots(self):   #on met les mots du document dans une liste
95          f = open('mots.txt','r')
96          s = f.read()
97          l = s.split('\n')
98          f.close()
99          return l
```

La méthode chargeMots permet de mettre tous les mots du fichier texte dans une liste.

4. Méthode affiche_pendu

```
102     def affiche_pendu(self):    #pour afficher l'image du pendu dans la zone d'affichage
103         photo=self.__photos[self.__nbmanques]
104         self.__zoneAffichage.create_image(0,0,anchor=NW, image=photo)
105         self.__zoneAffichage.config(height=photo.height(),width=photo.width())
```

Cette méthode permet d'afficher les images du pendu en fonction du nombre de coups ratés.

Dans le cas des formes, on a la méthode suivante :

5. Méthode affiche_formes

```
86      def affiche_forme(self):    #Affichage d'un bout du pendu
87          liste=[[30,200,150,205],[88,40,93,200],[88,40,180,45],[175,40,180,80],[163,75,193,105],[170,90,186,140],
    [150,110,178,115],[178,110,205,115],[170,120,175,170],[181,120,186,170]]
88
89          nb=self.__nbmanques-1
90          if nb== 4: # pour la tête
91              self.__zoneAffichage.create_oval(liste[nb][0],liste[nb][1],liste[nb][2],liste[nb][3],fill = "black")
92
93          else: # pour les rectangles
94              self.__zoneAffichage.create_rectangle(liste[nb][0],liste[nb][1],liste[nb][2],liste[nb][3],fill = "black")
```

Dans ce cas, on affiche des formes au fur et à mesure plutôt que d'afficher des images.

6. Méthode fin de partie

```
107     def finpartie(self):
108         if self.__nbmanques==10:  #la partie est perdue
109             a='Vous avez perdu, le mot était:'+self.__mot
110             self.__motaffiche.config(text=a)
111             self.affiche_pendu()     #on affiche le pendu
112             for i in range(1,27):
113                 self.__boutons[i-1].config(state=DISABLED)   #on désactive tous les boutons
114 
115             #gestion du score
116             fichier=open(self.__joueur+'.txt','r')
117             text=fichier.read()
118             a=(self.__mot,'échec')
119             with open(self.__joueur+'.txt','w') as f:
120                 f.write(str(text)+"\n"+str(a))
121                 f.close()
122 
123             fichier = open(self.__joueur+'.txt', "r")
124             text1=fichier.readline()
125             text2=fichier.readline()   #score
126             text3=fichier.readline()   #nbpartiesgagnées
127             text4=fichier.readline()    #nbpartiesjouées
128             text=fichier.read()
129 
130             text4=str(int(text4)+1)   #une partie jouée de plus
131             text2=str(int(text3)/int(text4)*100)    #Calcul du nouveau score
132 
133             with open(self.__joueur+'.txt','w') as f:
134                 f.write(str(text1)+str(text2)+"\n"+str(text3)+str(text4)+"\n"+str(text))
135                 f.close()
136 
137             fichier.close()
138 
139         if self.__mot==self.__motdecouvert:  #la partie est gagnée
140             a=self.__mot+ '-Bravo, vous avez gagné'
141             self.__motaffiche.config(text=a)
```

```
142             for i in range(1,27):
143                 self.__boutons[i-1].config(state=DISABLED)    #On désactive tous les boutons
144 
145             #gestion du score
146             fichier=open(self.__joueur+'.txt','r')
147             text=fichier.read()
148             a=(self.__mot,'victoire')
149             with open(self.__joueur+'.txt','w') as f:
150                 f.write(str(text)+"\n"+str(a))
151                 f.close()
152 
153             fichier = open(self.__joueur+'.txt', "r")
154             text1=fichier.readline()
155             text2=fichier.readline()   #score
156             text3=fichier.readline()   #nbpartiesgagnées
157             text4=fichier.readline()    #nbpartiesjouées
158             text=fichier.read()
159 
160             text4=str(int(text4)+1)   #une partie jouée de plus
161             text3=str(int(text3)+1)   #une partie gagnée de plus
162             text2=str(int(text3)/int(text4)*100)    #Calcul du nouveau score
163 
164             with open(self.__joueur+'.txt','w') as f:
165                 f.write(str(text1)+str(text2)+"\n"+str(text3)+"\n"+str(text4)+"\n"+str(text))
166                 f.close()
167 
168             fichier.close()
169 
```

La fonction fin de partie est appelée dans le cas d'une victoire ou d'une défaite. Tous les boutons sont désactivés. Dans le cas où le nombre d'échecs est 10, c'est une défaite et on affiche l'image finale du pendu. Dans l'autre cas, c'est une victoire.

On finit par calculer le nouveau score du joueur.

7. Méthodes liées aux boutons principaux

```
171     def affiche_score(self):
172         fichier = open(self.__joueur+'.txt', "r")
173         text1=fichier.readline()
174         text2=fichier.readline()   #score
175 
176         self.__score.config(text='Score :'+text2)
177 
178 
179     def affiche_historique(self):
180         fichier = open(self.__joueur+'.txt', "r")
181         fichier.readline()
182         fichier.readline()
183         fichier.readline()
184         fichier.readline()   #on a sauté les 4 premières lignes: nom, score, parties jouées, parties gagnées
185 
186         text=fichier.read()   #le fichier sans les 4 premières lignes
187         print(text)   #On affiche l'historique dans la console
188 
189 
190     def reinitialiser(self):    #on écrase le fichier du joueur
191         with open(self.__joueur+'.txt','w') as f:
192             f.write(self.__joueur+"\n"+'0.0'+"\n"+'0'+"\n"+'0')
```

Les fonctions permettent d'afficher le score du joueur dans la fenêtre, d'afficher l'historique dans la console et de réinitialiser le score et l'historique du joueur.

Pour la fonction réinitialiser, on écrase le fichier du joueur et on écrit son nom, 0 pour le score, 0 pou le nombre de parties jouées et gagnées.

8. Méthode Traitement

```
195     def traitement(self,lettre):
196         for i in range(len(self.__mot)):
197             if lettre==self.__mot[i]:  #la lettre est bien dans le mot à un moment
198                 self.__motdecouvert=self.__motdecouvert[:i]+self.__mot[i]+self.__motdecouvert[i+1:]
199                 self.__motaffiche.config(text='Mot :'+self.__motdecouvert)
200 
201         if self.__mot==self.__motdecouvert:  #la partie est gagnée
202             self.finpartie()
203 
204         elif lettre not in self.__mot :    #si la lettre n'est pas dans le mot
205             self.__nbmanques += 1
206             self.affiche_pendu()    #alors on affiche un bout du pendu
207 
208             if self.__nbmanques==10:  #la partie est perdue
209                 self.finpartie()
210 
```

Cette fonction est appelée lorsque l'on clique sur un bouton du clavier. Elle regarde si la lettre est dans le mot à deviner, si oui elle cette lettre dans le motAffiche qui est composé des l ettres ayant déjà été devinées ; si non elle affiche un bout du pendu. Dans le cas où ce clic était le 10ème échec elle appelle la fonction fin de partie.

Description des tests réalisés et des commentaires sur les résultats obtenus

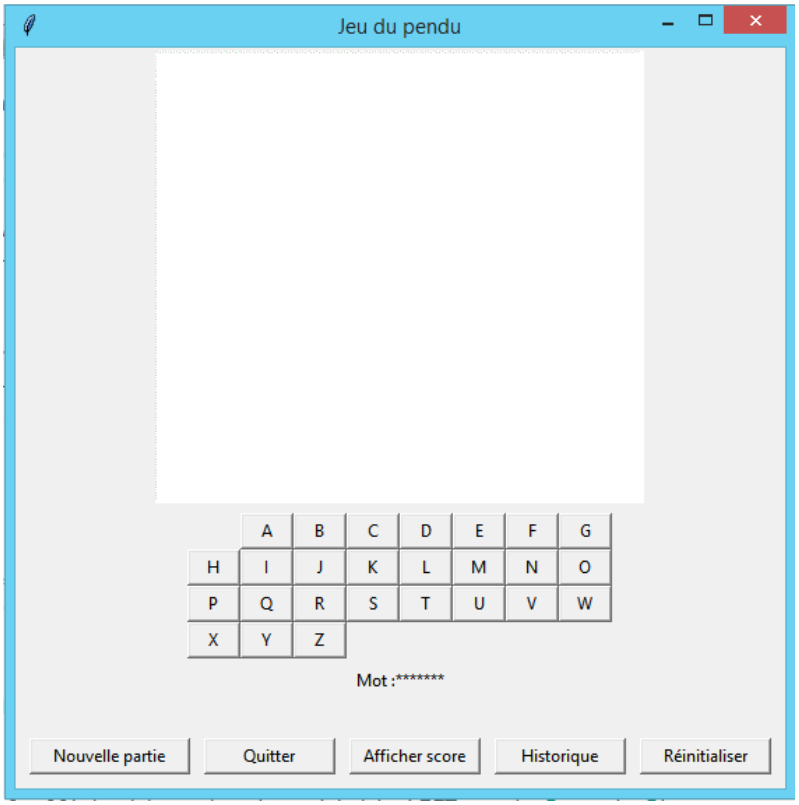
I. Fichier « pendu_images.py »

1. Exécution du script

```
Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 201
8, 13:32:41) on Windows (64 bits).
This is the Pyzo interpreter with integrated event
loop for TK.
Type 'help' for help, type '?' for a list of *magi
c* commands.
Running script: "C:\Users\Manon\Documents\Centrale
Lyon\INF TC2\TD5\pendu_images.py"
Tapez votre nom:
```

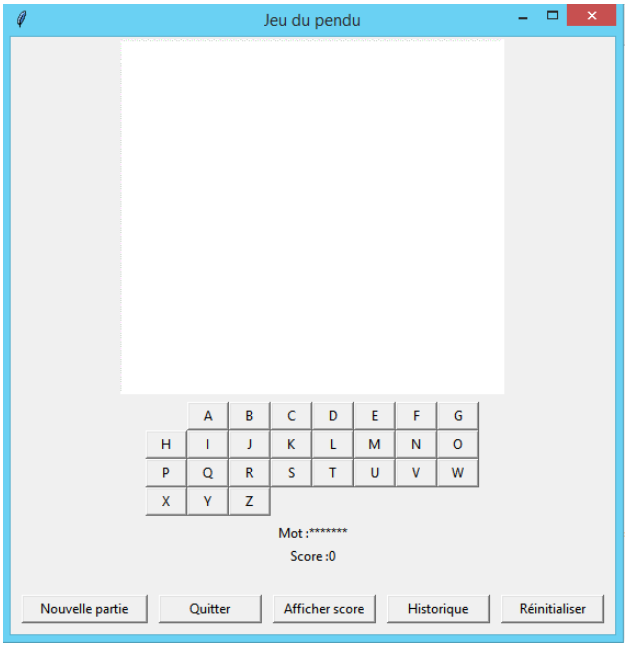
Tout d'abord, le jeu demande le nom du joueur pour identifier le fichier des scores et de l'historique des parties du joueur s'il existait déjà, ou pour créer ce fichier texte le cas échéant.

2. Début du jeu



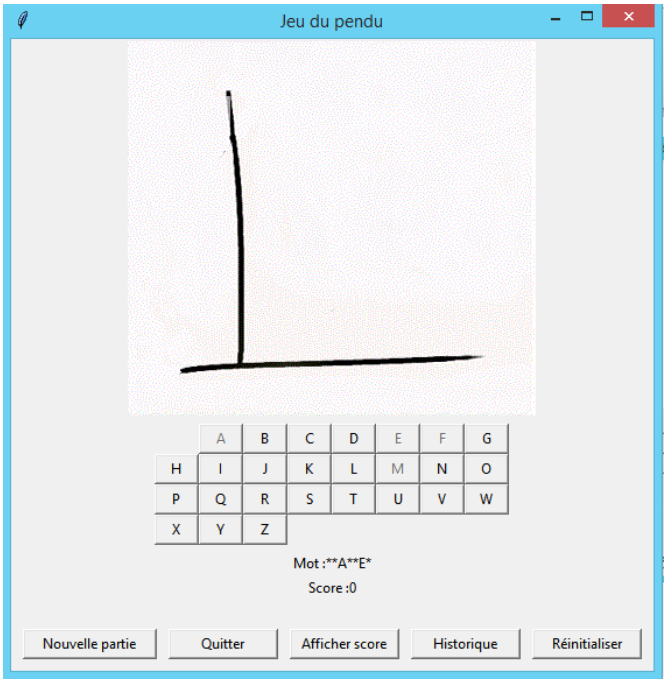
Une fois le nom du joueur entré, le jeu ouvre la fenêtre principale. Elle porte bien le nom de « jeu du pendu » et on y retrouve la zone d'affichage vierge, les boutons du clavier, le mot à deviner et les différents boutons.

3. Affichage du score initial



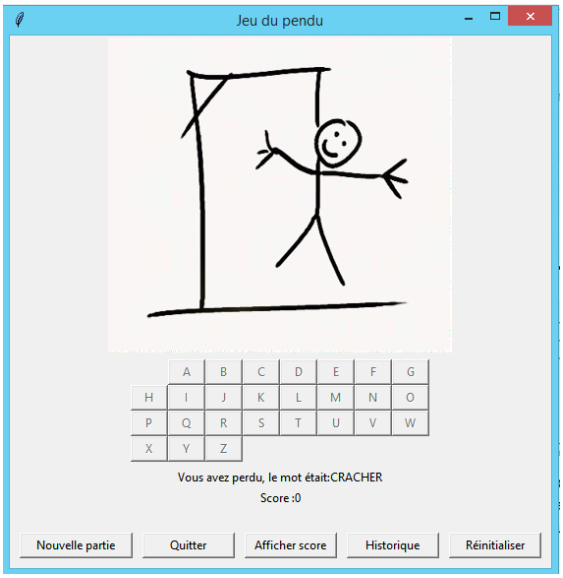
Lorsque l'on clique sur le bouton « Afficher score », le score du joueur s'affiche sous le mot à deviner.

4. Partie en cours



Le joueur a cliqué sur les lettres « A » et « E » qui étaient dans le mot, on les rend donc visibles. Le joueur a aussi cliqué sur les lettres « F » et « M » qui n'étaient pas dans le mot, l'image 'pendu2.gif' s'est donc affichée ; elle correspond au pendu lorsque deux fautes ont été faites.

5. Partie perdue



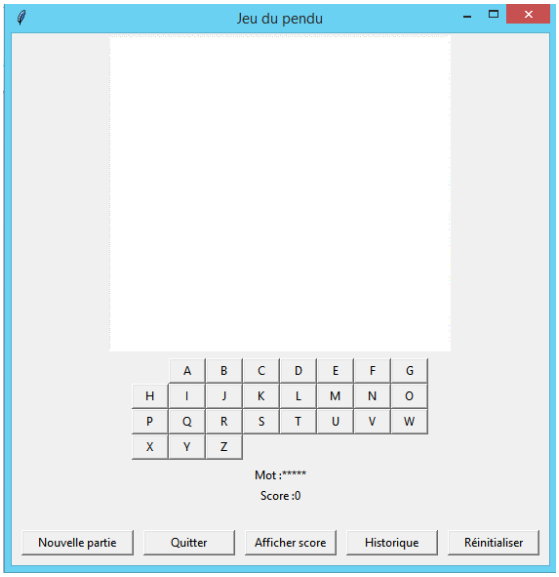
Dans ce cas, la partie a été perdue par le joueur. Le mot à deviner s'affiche en bas de la fenêtre, tous les boutons ont été désactivés et le score reste nul.

6. Affichage de l'historique

```
Running script: "C:\Users\Manon\Documents\Centrale
Lyon\INF TC2\TD5\pendu_images.py"
Tapez votre nom: Manon
Note: The GUI event loop is already running in the
pyzo kernel. Be aware
that the function to enter the main loop does not
block.
('CRACHER', 'échec')
```

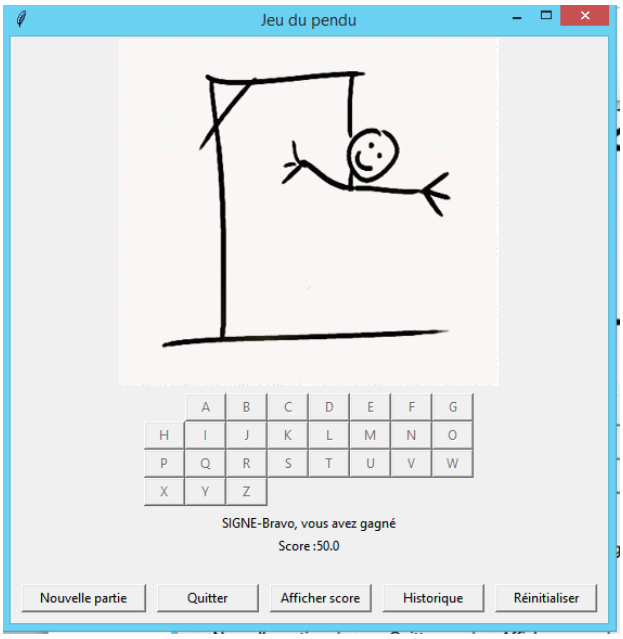
Lorsque l'on clique sur le bouton « Historique », le résumé des parties s'affiche dans la console avec dans chaque couple, le mot qu'il fallait deviner et l'issue de la partie.

7. Nouvelle partie

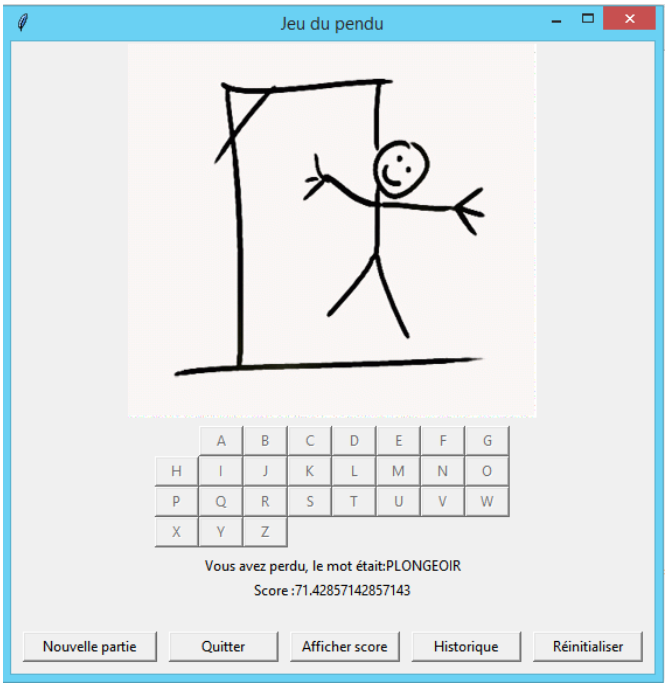


Le bouton « Nouvelle partie » réactive tous les boutons, sélectionne un nouveau mot au hasard dans la liste et efface tout dans la zone d'affichage.

8. Partie gagnée



9. Résultats au bout de quelques parties jouées



Historique :

```
('CRACHER', 'échec')
('CRACHER', 'échec')
('SIGNE', 'victoire')
('COIN', 'victoire')
('MAIN', 'victoire')
('ADROIT', 'victoire')
('TROUVER', 'victoire')
('PLONGEOIR', 'échec')

>>>
```

10. Réinitialisation

Lorsque l'on appuie sur le bouton « Réinitialiser », le score retombe à 0 et l'historique est à nouveau vide.

II. Fichier « pendu_formes.py »

Le principe de fonctionnement est le même, cependant ce ne sont pas des images qui s'affichent au fur et à mesure mais des formes.