



Compte-rendu d'informatique

INF tc2 TD5

Jeu du pendu

Ugo INSALACO

Année 2019-2020

Sommaire

| | | |
|----------|--|----------|
| 1 | Introduction | 2 |
| 2 | Diagrammes UML | 2 |
| 2.1 | MonBouton | 2 |
| 2.2 | ZoneAffichage | 2 |
| 2.3 | FenPrincipale | 2 |
| 3 | Code source commenté | 4 |
| 3.1 | MonBouton et ZoneAffichage | 4 |
| 3.2 | Initialisation de la fenêtre principale | 4 |
| 3.3 | Gestion du score | 5 |
| 3.4 | Gestion du nom du joueur | 6 |
| 3.5 | Affichage de l'historique et nouvelle partie | 6 |
| 3.6 | Traitement et gestion des mots | 6 |
| 3.7 | Fin de partie | 7 |
| 4 | Les test | 7 |
| 4.1 | Tests d'Initialisation | 7 |
| 4.2 | Test du clavier | 8 |
| 4.3 | Test d'une partie | 8 |
| 4.4 | Test sur le score | 9 |

1 Introduction

Ce rapport présente le travail réalisé dans le but de recréer un jeu du pendu classique sur python grâce à la bibliothèque tkinter. Le but est simple : deviner un mot caché en proposant des lettres et en faisant moins de 10 erreurs. A la fin de sa partie, le joueur doit pouvoir recommencer une partie avec un nouveau mot.

Les parties jouées sont enregistrées dans un document texte et chaque joueur a accès à son ratio de parties gagnées sur parties total. Il peut aussi à tout instant réinitialiser ce ratio ou afficher les informations de chaque partie qu'il a joué auparavant.

2 Diagrammes UML

2.1 MonBouton

La classe MonBouton intervient dans la création du clavier du jeu qui permet de proposer une lettre. Elle hérite de la classe Button de tkinter et ajoute comme fonctionnalité le stockage d'un caractère (qui sera la touche représentée) et une fonction spéciale appelée lors du clique qui sera utilisée par la fenêtre principale pour la logique du jeu.

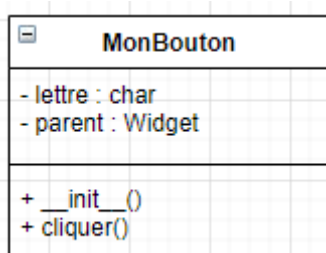


FIGURE 2.1 – Diagramme UML de la classe MonBouton

2.2 ZoneAffichage

La classe ZoneAffichage est l'endroit où l'on voit dessiné le pendu en fonction du nombre d'erreurs du joueur. Cette classe hérite de Canvas dans tkinter et a comme fonction de récupérer les images du pendu dans le dossier 'Images', de les stocker et de pouvoir changer l'image actuelle affichée. Les images stockées sont des objets PhotoImage de tkinter.

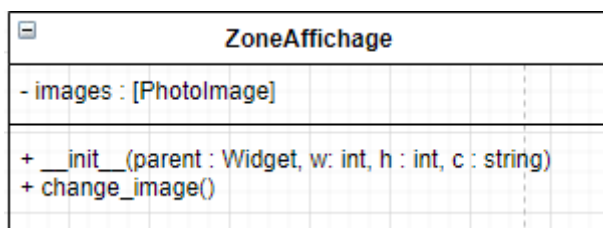


FIGURE 2.2 – Diagramme UML de la classe ZoneAffichage

2.3 FenPrincipale

FenPrincipale est la classe la plus complexe du programme puisqu'elle définit toute la logique du jeu et les interactions avec l'utilisateur. Cette classe possède 26 instances de MonBouton stockées dans la liste boutons, qui permettent de recréer un clavier. Elle possède aussi une ZoneAffichage pour les images du pendu. certains boutons comme 'Nouvelle Partie' ou 'Quitter' ne sont pas dans le diagramme UML car ils sont simplement créés lors de la fonction 'init' et plus jamais réutilisés.

Les champs 'nom', 'bouton-envoi' ou 'label-nom' permettent de mettre en place l'entrée du nom de l'utilisateur pour débiter le jeu. Quant aux champs 'score' ou 'label-score' permettent la gestion du score.

Le champs 'playing' à été mis en publique car la classe MonBouton a besoin de sa valeur pour savoir si le clavier doit être activé ou non (En début de partie le joueur doit d'abord entrer son nom et ne doit donc pas pouvoir interagir avec le clavier). Une alternative à mettre ce champs en publique aurait été de créer une méthode get pour ce champs uniquement et de l'appeler dans la classe MonBouton

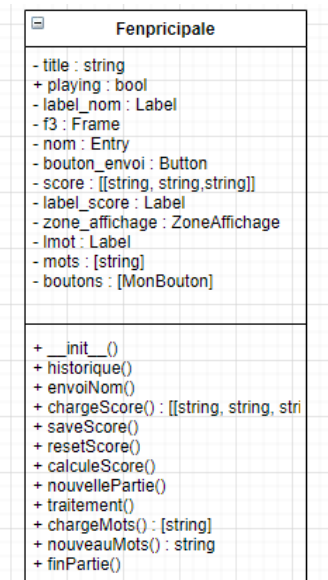


FIGURE 2.3 – Diagramme UML de la classe FenPrincipale

Voici finalement le diagramme des interactions entre les différentes classes :

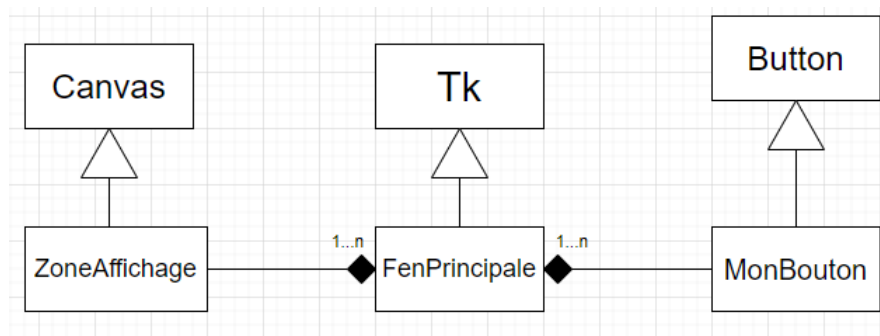


FIGURE 2.4 – Diagramme d'héritage

3 Code source commenté

3.1 MonBouton et ZoneAffichage

```
183
184 class ZoneAffichage(Canvas):
185     def __init__(self, parent, w, h, c):
186         Canvas.__init__(self, width = w, height = h, bg = c)
187         self.__images = []
188         chemin = 'ImagesPendou/pendu' #chemin d'accès aux images
189
190         #Recupération de chaque image du pendu
191         for i in range(1,9,1):
192             self.__images.append(PhotoImage(file=chemin + str(i) + '.gif'))
193
194             self.create_image(0,0, anchor=NW, image=self.__images[0]) #affichage de la première image
195             #dans la zone d'affichage
196
197     def change_image(self, i):
198         self.delete('All') #on supprime l'image actuelle
199         self.create_image(0,0, anchor=NW, image=self.__images[i]) #on la remplace par une nouvelle
200
201 class MonBouton(Button):
202     def __init__(self, frame, lettre, parent):
203         Button.__init__(self, frame, text = lettre, command = self.cliquer)
204         self.__lettre = lettre
205         self.__parent = parent
206
207     def cliquer(self):
208         if self.__parent.playing :
209             self.config(state = "disabled") #on désactive le bouton
210             self.__parent.traitement(self.__lettre) #on envoie le traitement à la fenêtre parente
211         return
212
```

FIGURE 3.1 – Code source des classes ZoneAffichage et MonBouton

Ce code permet l'implémentation des classes ZoneAffichage et MonBouton

3.2 Initialisation de la fenêtre principale

```
7 from tkinter import*
8 from random import*
9
10
11 class FenPrincipale(Tk):
12     def __init__(self):
13         Tk.__init__(self)
14         self.__title = "Jeu du pendu"
15
16         #boutons nouvelle partie et quitter
17         f1 = Frame(self) # on crée une frame
18         f1.pack(side=TOP, padx=5, pady=5)
19         #on crée les boutons et on les ajoute à la frame
20         Button(f1, text = 'Nouvelle partie', width=15, command = self.nouvellePartie).pack(side=LEFT, padx = 5, pady = 5)
21         Button(f1, text = 'Quitter', width = 15, command = self.destroy).pack(side = LEFT, padx = 5, pady = 5)
22         Button(f1, text = 'Reset Score', width = 15, command = self.resetScore).pack(side = LEFT, padx = 5, pady = 5)
23
24
25         #Entrée du nom
26         self.playing = False # booléen représente si l'on est en train de jouer la partie
27         self.__sv = StringVar()
28         self.__label_nom = Label(self)
29         self.__label_nom.config(text = 'Veuillez entrer votre nom')
30         self.__label_nom.pack(side = TOP, padx = 5, pady = 5)
31         self.__f3 = Frame(self)
32         self.__f3.pack(side = TOP, padx = 5, pady = 5)
33         self.__nom = Entry(self.__f3, text = 'Entrez votre nom', textvariable = self.__sv)
34         self.__nom.pack(side = LEFT, padx = 5, pady = 5)
35         self.__bouton_envoi = Button(self.__f3, text = 'Envoyer', width = 15, command = self.envoiNom)
36         self.__bouton_envoi.pack(side = LEFT, padx = 5, pady = 5)
37
```

```

38     #gestion du score
39     self.__score = self.chargeScore()
40     self.__label_score = Label(self.__f3)
41
42     #zone affichage
43     self.__zone_affichage = ZoneAffichage(self, 350,320,'white')
44     self.__zone_affichage.pack(padx=5, pady=5)
45
46     #Label mot
47     self.__lmot = Label(self)
48     self.__lmot.pack(side=TOP)
49
50     #gestion des mots
51     self.__mots = self.chargeMots()
52
53     #boutons clavier
54     self.__boutons = []
55     f2 = Frame(self)
56     for i in range(26):
57         button = MonBouton(f2, chr(ord('A')+i), self)
58         button.grid(row = (i//7)+1, column = i%7 + (i//7)//3)
59         self.__boutons.append(button)
60     f2.pack()
61
62     #bouton historique
63     Button(f1, text = 'Historique', width = 15, command = self.historique).pack(side = LEFT, padx = 5, pady = 5)
64     #Lancement partie
65     self.nouvellePartie()
66

```

FIGURE 3.2 – Code source de la classe FenPrincipale (initialisation)

3.3 Gestion du score

```

78 def chargeScore(self):
79     with open('score.txt', 'r') as f: #on ouvre le fichier score
80         l = f.read().splitlines()      #on lise les lignes
81         for i in range(len(l)):
82             l[i] = l[i].split(',') #on range les informations dans un tableau de tableau
83         return l
84
85 def saveScore(self):
86     with open('score.txt', 'w') as f: #on ouvre le fichier score
87         for t in self.__score:
88             f.write(','.join(t)+'\n') #on écrit chaque ligne de la liste score dans le fichier score
89
90 def resetScore(self):
91     l = []
92     if self.playing: #on vérifie que le joueur a déjà entré son prénom
93         for t in self.__score:
94             if t[0] != self.__nom_joueur: #on vérifie toutes les parties que le joueur n'a pas joué
95                 l.append(t) #on les ajoute dans la nouvelle liste
96         self.__score = l
97         print('Score reset')
98         self.saveScore() #on enregistre le nouveau score
99         self.__player_score = self.calculeScore() #on calcule le nouveau score du joueur
100         self.__label_score.config(text = 'Votre score actuel est de : '+str(self.__player_score)+' %')
101     else:
102         print('/!\ Le joueur n\'est pas encore défini')
103
104 def calculeScore(self):
105     p = 0 #nombre de parties jouées par le joueur
106     v = 0 #nombre de victoires du joueur
107     for t in self.__score: #on parcourt les parties
108         if t[0] == self.__nom_joueur: #on regarde celles que le joueur actuel a joué
109             if t[2] == 'True': #on regarde celles qu'il a gagnées
110                 v+=1
111             p+=1
112     if p!=0: #si le joueur n'a joué aucune partie
113         return v/p*100
114     else:
115         return 0

```

FIGURE 3.3 – Code source de la classe FenPrincipale (gestion du score)

3.4 Gestion du nom du joueur

```
67 def envoiNom(self):
68     self.__nom_joueur = self.__sv.get() #on récupère Le nom du joueur
69     self.__nom.pack_forget() #on supprime l'Entry du nom du joueur
70     self.__bouton_envoi.pack_forget() #on supprime Le bouton d'envoi du nom du joueur
71     self.__label_nom.pack_forget() # on Supprime Le Label nom du joueur
72     self.playing = True #on met la partie comme étant en cours
73     self.__player_score = self.calculScore() #on Calcule Le score du joueur
74     self.__label_score.config(text = 'Votre score actuel est de : '+str(self.__player_score)+' %')
75     self.__label_score.pack(side = LEFT, padx = 5, pady = 5)
76     return
```

FIGURE 3.4 – Code source de la classe FenPrincipale (gestion du joueur)

3.5 Affichage de l'historique et nouvelle partie

```
117 def historique(self):
118     if self.playing:
119         print('Historique de tes parties :')
120         for t in self.__score: #On parcourt la liste des parties et on affiche celle du joueur actuel
121             if t[0] == self.__nom_joueur:
122                 print('mot : '+t[1]+ ' , Gagné : '+t[2])
123             else :
124                 print('/!\ Le joueur n'est pas défini')
125
126 def nouvellePartie(self):
127     #gestion du mot
128     self.__mot = self.nouveauMot() #on choisit un nouveau mot
129     self.__motAffiche = '*'*len(self.__mot) #on reset le mot a afficher
130     self.__lmot.config(text=self.__motAffiche)
131
132     #gestion des boutons
133     for i in range(26):
134         self.__boutons[i].config(state = 'active') #on remet actif chaque bouton du clavier
135
136     #gestion du jeu
137     self.__nbManques = 0 #on réinitialise Le nombre d'erreurs
138     self.__zone_affichage.change_image(0) #on remet le pendu à 0
139
```

FIGURE 3.5 – Code source de la classe FenPrincipale (historique et nouvelle partie)

3.6 Traitement et gestion des mots

```
140 def traitement(self, lettre):
141     b = False
142     for i in range(len(self.__mot)): #on parcourt Les lettres du mot recherché
143         if self.__mot[i] == lettre:
144             l = list(self.__motAffiche)
145             l[i] = lettre #on change la lettre du mot à afficher si elle est bonne
146             self.__motAffiche = "".join(l)
147             b = True
148             self.__lmot.config(text=self.__motAffiche) # on réaffiche le nouveau motAffiche
149
150     if self.__motAffiche == self.__mot: # si le mot entièrement trouvé on fini la partie
151         self.finPartie(True)
152
153     if not b: #si on a proposé une mauvaise lettre
154         self.__nbManques+=1
155         if self.__nbManques < 8 : #s'il nous reste des essais à faire
156             self.__zone_affichage.change_image(self.__nbManques)
157         if self.__nbManques == 7: #si l'on a plus d'essais
158             self.finPartie(False)
159
160 def chargeMots(self):
161     with open('mots.txt','r') as f:
162         l = f.read().splitlines() #on stock les mots du fichier mot dans un tableau
163     return l
164
165 def nouveauMot(self):
166     return choice(self.__mots) #on choisit un mot dans la liste de mots
167
```

FIGURE 3.6 – Code source de la classe FenPrincipale (traitement et gestion des mots)

3.7 Fin de partie

```
167
168 def finPartie(self, b):
169     for i in range(26):
170         self.__boutons[i].config(state = 'disabled') #on désactive tous les boutons du clavier
171
172     if b :
173         self.__lmot.config(text='Bravo tu as trouvé le mot')
174     else:
175         self.__lmot.config(text='Dommage tu as perdu, le mot était : '+self.__mot)
176
177     self.__score.append([self.__nom_joueur, self.__mot, str(b)]) #on ajoute la partie à la liste des parties
178     self.saveScore() #on sauvegarde le nouveau score
179     self.chargeScore() #on récupère le score
180     self.__player_score = self.calculScore()
181     self.__label_score.config(text = 'Votre score actuel est de : '+str(self.__player_score)+' %')
182
183
```

FIGURE 3.7 – Code source de la classe FenPrincipale (fin de partie)

4 Les test

Les tests permettent de mettre en évidence certains problèmes dans le code ou certaines erreurs dans les fonctionnalités. Ils sont ici réalisés à partir d’une copie du code source à laquelle on a rajouté des fonctions d’affichage dans la console pour savoir plus en détail où en est le programme.

4.1 Tests d’Initialisation

Le premier test à réaliser est de vérifier que le programme se lance bien avec chaque Widget à sa place.

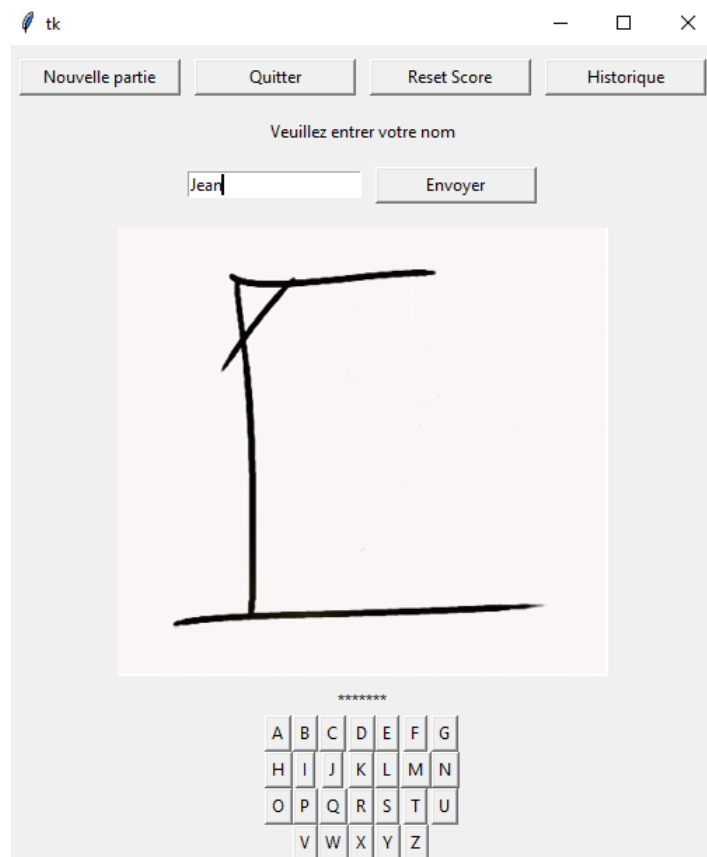


FIGURE 4.1 – Test d’affichage de la fenêtre de jeu

Ensuite, il est nécessaire de vérifier que les mots présents dans le fichier 'mots' s’importent bien et qu’il est possible d’en récupérer un aléatoirement.


```
'GARE', 'GRUE', 'HELICOPTERE', 'MOTO', 'PANNE', 'PARKING',
'PILOTE', 'PNEU', 'QUAI', 'TRAIN', 'VIRAGE', 'VITESSE',
'VOYAGE', 'WAGON', 'ZIGZAG', 'ARRETER', 'ATTERRIR',
'BOUDER', 'CHARGER', 'CONDUIRE', 'DEMARRER',
'DISPARAITRE', 'DONNER', 'ECRASER', 'ENVOLER', 'GARDER',
'GARER', 'MANQUER', 'PARTIR', 'POSER', 'RECULER',
'ROULER', 'TENDRE', 'TRANSPORTER', 'VOLER', 'ABIME',
'ANCIEN', 'BLANC', 'BLEU', 'CASSE', 'CINQ', 'DERNIER',
'DEUX', 'DEUXIEME', 'DIX', 'GRIS', 'GROS', 'HUIT',
'JAUNE', 'MEME', 'NEUF', 'PAREIL', 'PREMIER', 'QUATRE',
'ROUGE', 'SEPT', 'SEUL', 'SIX', 'SOLIDE', 'TROIS',
'TROISIEME', 'UN', 'VERT', 'DESSUS', 'AUTOUR', 'VITE',
'VERS', 'ACROBATE', 'ARRET', 'ARRIERE', 'BARRE',
'BARREAU', 'BORD', 'BRAS', 'CERCEAU', 'CHAISE',
```

FIGURE 4.2 – Test d’importation des mots

Ici on observe bien que la liste des mots est affichée dans la console du programme.

De même, on peut vérifier que la liste des images est bien importée.

4.2 Test du clavier

On teste ensuite chaque bouton du clavier afin de vérifier que la classe FenPrincipale reçoit les bonnes lettres et est capable de les traiter correctement.

```
Le bouton C a été cliqué
Envoi de la lettre C à la fenêtre principale
La lettre C a été reçu
La lettre n'était pas bonne
Le nombre d'erreurs actuel est de : 1
```

FIGURE 4.3 – Test du clavier (affichage dans la console)

On peut vérifier que le bouton correspondant à la lettre U est bien désactivé ensuite

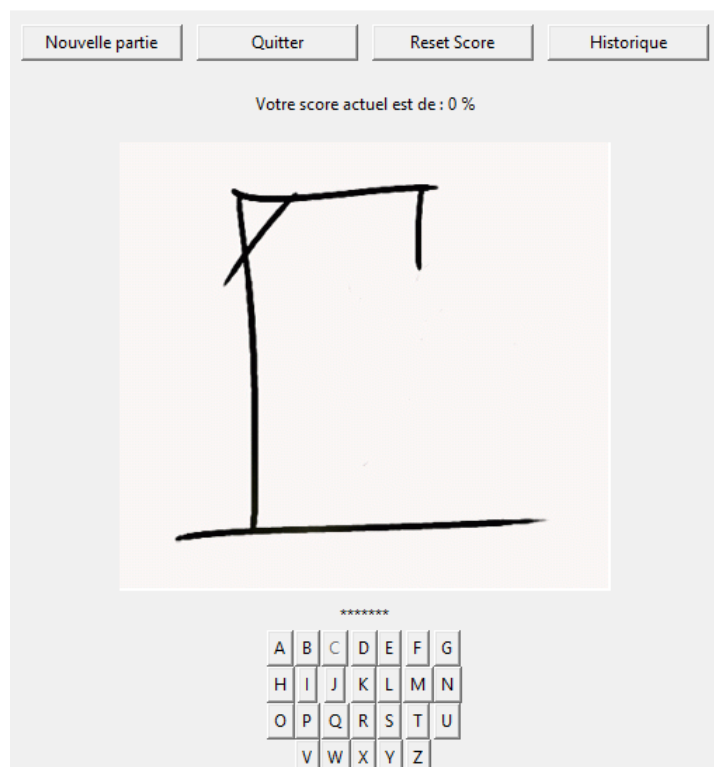


FIGURE 4.4 – Test du clavier (bouton U désactivé)

4.3 Test d’une partie

Avant de lancer une partie le joueur doit entrer son nom. On peut vérifier que le nom est bien récupéré en ajoutant un print dans la fonction 'envoiNom'. On peut ensuite jouer une partie et vérifier que les victoires et défaites se déroulent comme prévu.

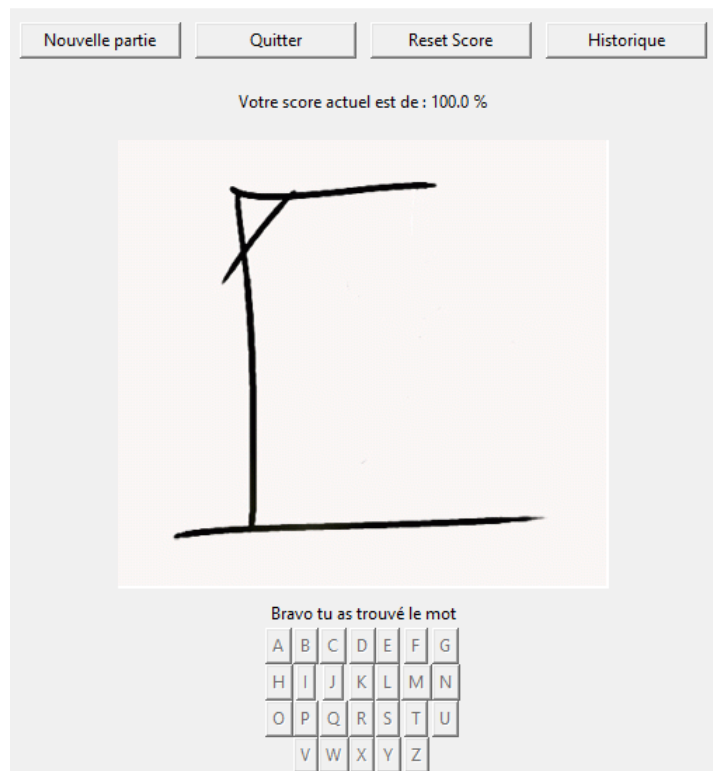


FIGURE 4.5 – Test de la victoire

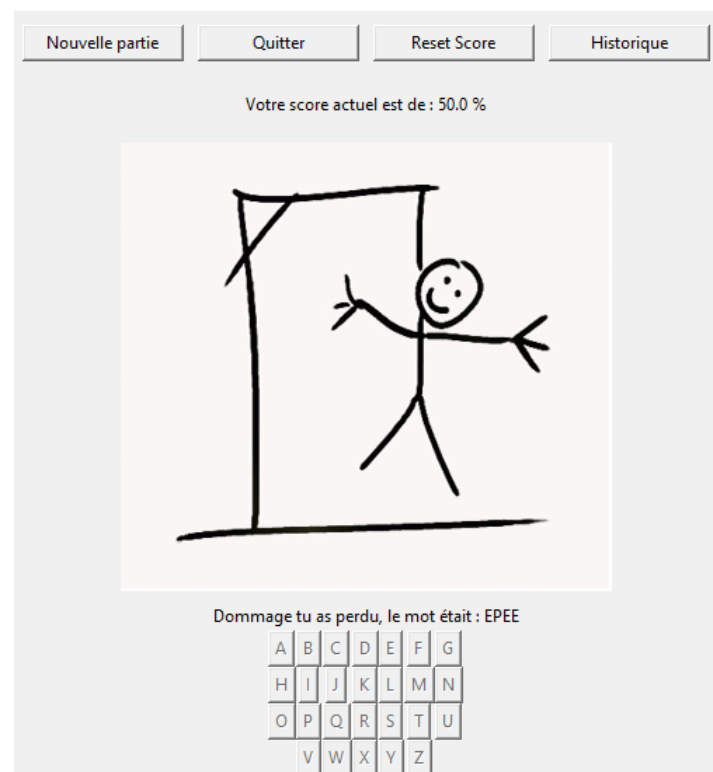


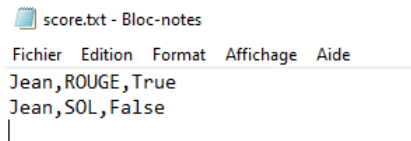
FIGURE 4.6 – Test de la défaite

On peut enfin relancer une partie pour voir si tout se déroule comme prévu.

4.4 Test sur le score

La dernière chose à tester est le fonctionnement du système de score.

On teste tout d'abord le chargement des parties. Le fichier des parties de départ est vide puis après avoir fait deux parties on obtient le suivant :



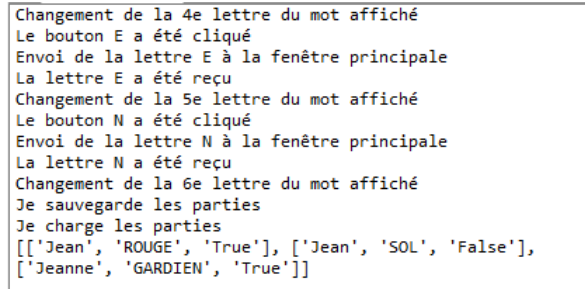
```

score.txt - Bloc-notes
Fichier Edition Format Affichage Aide
Jean,ROUGE,True
Jean,SOL,False
|

```

FIGURE 4.7 – Premier fichier score enregistré

On effectue ensuite une partie avec un autre joueur et dès la fin de la partie le fichier score est réactualisé

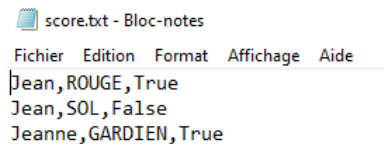


```

Changement de la 4e lettre du mot affiché
Le bouton E a été cliqué
Envoi de la lettre E à la fenêtre principale
La lettre E a été reçu
Changement de la 5e lettre du mot affiché
Le bouton N a été cliqué
Envoi de la lettre N à la fenêtre principale
La lettre N a été reçu
Changement de la 6e lettre du mot affiché
Je sauvegarde les parties
Je charge les parties
[['Jean', 'ROUGE', 'True'], ['Jean', 'SOL', 'False'],
['Jeanne', 'GARDIEN', 'True']]

```

FIGURE 4.8 – Console après la troisième partie



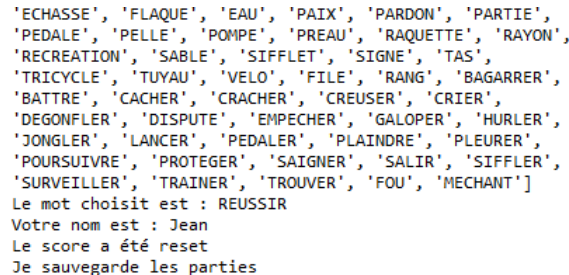
```

score.txt - Bloc-notes
Fichier Edition Format Affichage Aide
Jean,ROUGE,True
Jean,SOL,False
Jeanne,GARDIEN,True

```

FIGURE 4.9 – Fichier score après la troisième partie

On se replace enfin en tant que Jean pour tester la ré initialisation du score :

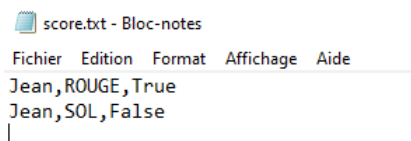


```

'ECHASSE', 'FLAQUE', 'EAU', 'PAIX', 'PARDON', 'PARTIE',
'PEDALE', 'PELLE', 'POMPE', 'PREAU', 'RAQUETTE', 'RAYON',
'RECREATION', 'SABLE', 'SIFLET', 'SIGNE', 'TAS',
'TRICYCLE', 'TUYAU', 'VELO', 'FILE', 'RANG', 'BAGARRER',
'BATTRE', 'CACHER', 'CRACHER', 'CREUSER', 'CRIER',
'DEGONFLER', 'DISPUTE', 'EMPECHER', 'GALOPER', 'HURLER',
'JONGLER', 'LANCER', 'PEDALER', 'PLAINdre', 'PLEURER',
'POURSUIVRE', 'PROTEGER', 'SAIGNER', 'SALIR', 'SIFFLER',
'SURVEILLER', 'TRAINER', 'TROUVER', 'FOU', 'MECHANT']
Le mot choisit est : REUSSIR
Votre nom est : Jean
Le score a été reset
Je sauvegarde les parties

```

FIGURE 4.10 – Console après ré initialisation du score de Jean



```

score.txt - Bloc-notes
Fichier Edition Format Affichage Aide
Jean,ROUGE,True
Jean,SOL,False
|

```

FIGURE 4.11 – Fichier score après ré initialisation du score de Jean