LA SOURCE

DOSSIER FINAL DE BAC

Terminale S, Spécialité ISN | Léa Cohen et Naomie Pereira

SOMMAIRE

Présentation

Cette année 2017-2016 de terminale Scientifique nous avons choisi la spécialité Informatique et Sciences du numérique car nous savions que l'informatique est aujourd'hui présent dans tous les domaines et prend de plus en plus d'importance. De plus, n'ayant auparavant aucune base dans le codage informatique, cela nous intéressait d'en découvrir les facettes. Notre groupe se compose de Léa COHEN et Naomie PEREIRA.

But du jeu : Notre jeu se présente sous la forme d'une enquête dans laquelle le joueur essaie de retrouver une personne disparue. Nous avons alors eu l'idée que le joueur, prenant la place d'un dénommé Paul, guiderait son ami Rose par SMS pendant que celle-ci interrogeait des personnes sur le terrain. Le but est à la fin de retrouver leur meilleur ami Kevin, disparu. Notre jeu se basant sur un arbre de choix, les réponses que le joueur choisit influent sur l'issue du jeu.

Nous avons inventé de A à Z le scénario et les issues de l'enquête.

Recherche d'idées et analyse des besoins

Nous voulions créer un jeu qui n'avait pas été souvent présenté comme projet de bac en spécialité ISN. Nous avons alors choisi de créer un text-based game. L'idée nous est venue en regardant une vidéo d'un youtuber qui jouait à un jeu appelé *Bury me, My Love*. Dans ce genre de jeu, le joueur se comporte en quelque sorte comme le héros d'une histoire qu'il vit à travers une suite de messages. Nous avons tout de suite adoré ce concept et le trouvions très intéressant.

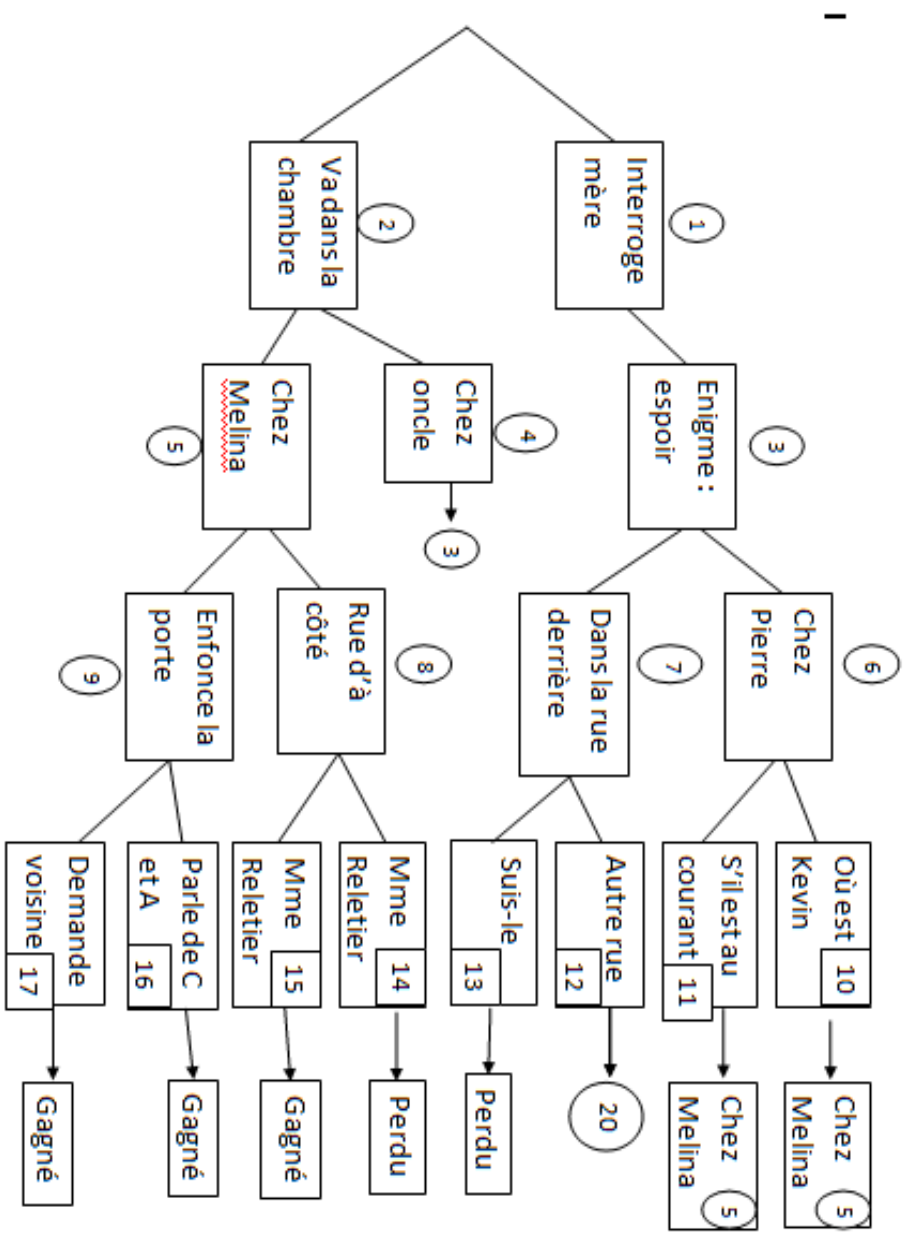
La première difficulté de coder ce genre de jeu est tout d'abord, même si cela ne demande pas de programmation, de créer un scénario cohérent qui sera la base de notre jeu. Celui-ci se présente sous la forme d'un arbre de décision. Dans le code, celui-ci se présente sous la forme de conditions (if, elif). De plus, lorsque nous cherchions un modèle ou un code sur ce type de jeu, dans la majorité des cas, on trouvait de nombreux programmes déjà fait et la seule partie modifiable était le texte. Nous avons donc cherché plusieurs bibliothèques graphiques jusqu'à trouver Tkinter. En parallèle nous nous documentions souvent sur le langage python et ses fonctionnalités. Nous avons ainsi utilisé :

- Des sites sur le langage python : http://www.france-ioi.org/ pour apprendre les bases de python, http://inforef.be/swi/download/apprendre_python3.pdf pour connaitre plus en détail
- Des sites pour comprendre le fonctionnement de Tkinter : http://apprendre-python.com/page-tkinter-interface-graphique-python-tutoriel , http://tkinter.fdex.eu/index.html
- l'interface Tkinter
- Pycharm qui est un environnement de développement spécialisé dans le langage Python.
- Google drive pour échanger nos idées et avancer plus rapidement dans le projet
- Les modules python comme tkinter, pygame et time.
- Audacity pour l'enregistrement audio

Structuration du projet

Pour créer notre projet, Nous avons inventé de A à Z le scénario et les issues de l'enquête. Il y a deux fins possibles : soit le joueur gagne, soit le joueur perd.

NOTRE ARBRE DE DÉCISIONS : (il ne montre que les choix de réponses du joueur et non les dialogues qui s'enchaînent entre les réponses choisies par le joueur)



FONCTIONNEMENT GLOBAL DU JEU :

Le jeu s'ouvre sur une fenêtre d'accueil avec un bouton "jouer". Lorsque le joueur clique sur le bouton, il va alors lancer un enregistrement qui permet d'expliquer le contexte et le but du jeu. Ensuite, une nouvelle fenêtre s'ouvre où l'on découvre l'interface de messages. Nous avons numéroté chaque message pour faciliter leur déroulement par la suite.

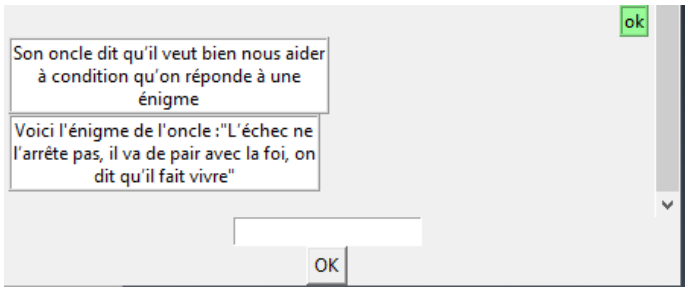
Les 7 premiers messages étant automatiques, il suffisait de lier ces messages en passant au suivant avec : `message_a_afficher += 1`

Ensuite lorsque vient une question, il faut alors mettre des boutons radios sur lesquels le joueur va cliquer et à côté les propositions de réponses. Toutes les réponses sont associées à des variables dans un dictionnaire. On avait au départ placé les réponses dans une liste mais le dictionnaire nous a paru par la suite plus pratique.

```
#création d'un dictionnaire contenant toutes les réponses aux questions à choix
liste_reponse={ 'r1' : 'Interroge sa mère pour savoir où il est allé en premier',
                'r2' : 'Tente de trouver des indices dans sa chambre',
                'r6' : 'Celui qui dit que Kevin est allé chez Pierre, un ami à lui dans une rue à droite',
                'r7' : 'Celui qui dit que Kevin est allé voir un de ses partenaires de travail dans une rue à gauche',
                'r10' : 'où Kevin est allé après être allé le voir',
                'r11' : 's\'il est au courant de quelque chose de louche qui pourrait être liée à la disparition de Kevin',
                'r12' : 'c\'est louche, on retourne dans la rue de droite',
```

En utilisant un système de conditions "if/elif" lorsque le joueur clique sur la réponse qu'il souhaitait, le message suivant apparaissait grâce à notre système de numérotation des messages. Et ainsi de suite.

Nous avons également inséré une énigme. Le joueur doit ainsi taper avec son clavier la réponse qu'il pense être la bonne dans une case vide. Nous avons dû définir au préalable les réponses considérées comme bonnes.

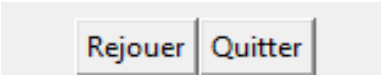



```
# mettre le champ de texte pour réponse
reponse_enigme = Entry(reponse_canvas)
```

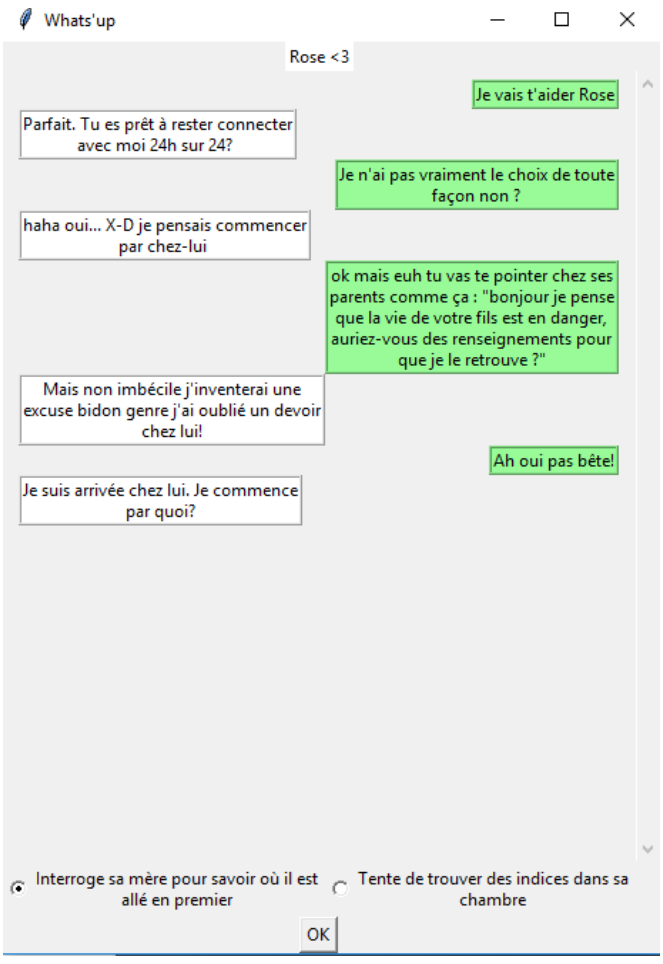
Cela ajoutait une petite difficulté car il fallait se mettre dans la peau d'un joueur lambda et se dire qu'il pouvait écrire la réponse avec une faute d'orthographe ou en ajoutant des espaces ou encore en écrivant certaines lettres en majuscules. Nous avons donc utilisé des fonctions :

```
# transforme tout ce que le joueur écrit en minuscule
reponse = reponse_enigme.get().lower()
# supprime les espaces avant et après un mot
reponse = reponse.strip()
```

S'en suivent les séries de messages, réponses et questions et vers la fin du jeu, le joueur a le choix entre "rejouer" ou "quitter" en cliquant sur un des boutons. S'il choisit rejouer alors le jeu recommence à partir du premier message. En effet puisque la messagerie du début est un peu longue et que le joueur l'a déjà entendu une fois, alors il connait déjà l'intro et le but du jeu. Cela ne nous semblait donc pas utile de lui faire écouter à nouveau.



Voici ce à quoi ressemble notre jeu (sur l'image on voit la première question impliquant le premier choix):



Répartition des tâches, démarche collaborative

Date	Tâches	Noms
12/01/18	**On avance sur le scénario de notre enquête en essayant de trouver des solutions pour une meilleure interactivité avec le joueur**	Léa
19/01	**On cherche les différents évènements qui mèneront à chacune des 3 fins du jeu**	Léa
26/01	**Création d'un arbre de choix (finalement il y aura deux fins possibles : gagner ou perdre)**	Léa et Naomie
02/02	**Recherche sur le nom du type de notre jeu**	Naomie
16/02	**On trouve la bibliothèque Tkinter.**	Naomie
19/02	**Renseignements sur python et Tkinter**	Léa et Naomie
22/02	**Début du code pour la création de la page d'accueil de notre jeu.**	Léa
28/02	**Création de la deuxième "fenêtre" de notre jeu**.	Naomie
15/03	**Résolution du problème de décalage entre le son et l'image**	Léa et Naomie

	messagerie	
16/03	**Réglage de mise en forme de notre fenêtre sur Tkinter**	Léa et Naomie
23/03	**Avancement du code et recherches**	Léa et Naomie
30/03	**Avancement du code et difficultés pour l'envoyer sur github**	Léa et Naomie
06/04	**Avancement du code et résolution du problème avec Github**	Léa et Naomie
13/04	**Avancement du code et difficultés rencontrées concernant le laps de temps entre l'affichage des messages**	Léa et Naomie
23/04	**Résolution du problème de laps de temps entre les messages et avancement du code (création des boutons radios)**	Léa et Naomie
25/04	**On continue notre code principal (suite de l'arbre de choix)**	Léa et Naomie
26/04	**On continue le code principal en rencontrant quelques difficultés concernant la Scrollbar et la numérotation de l'arbre de choix**	Léa et Naomie
01/05	**on continue notre arbre de choix, revoyons sa numérotation et finissons notre code (création d'un bouton "rejouer" et "quitter")**	Léa et Naomie
08/05	**On finalise notre code (commentaires etc…)**	Léa et Naomie

Nous avons fait presque tout ensemble. En effet pour les premières recherches d'idées nous nous sommes vues régulièrement. Puis pendant que Léa créait le scénario, Naomie cherchait les premiers éléments de code.

Puis pour l'échange et le partage du code, nous avons utilisé plusieurs plateformes de partages de données comme Github et Drive.

Github est une plateforme de partage très utilisée par les codeurs informatiques. Elle permet de prévoir une organisation des tâches avec différentes colonnes (planning, outils …) et de déposer le code que l'un ou l'autre a fait. Ainsi lorsque l'une déposait du code l'autre pouvait coder à tout moment quand elle était disponible pour continuer sur la version précédente. Cependant cette plateforme présente quelques limites puisque deux personnes ne peuvent pas coder simultanément sur le même fichier de code. De plus nous avons rencontré de nombreuses difficultés pour lier notre fichier comportant le code à la plateforme Github. De plus, nous avons constaté quelques bug sur Github. Par exemple, parfois, un message que nous avions écrit apparaissait en anglais sur notre journal puis lorsque l'on cliquait dessus pour le modifier, il réapparaissait en français...

Réalisation personnelle

Tout d'abord, dès que l'idée du projet nous est venue à l'esprit, j'ai écrit le scénario. J'ai dû le raccourcir plusieurs fois pour ne pas que l'arbre de choix devienne trop complexe au niveau des branches car nous savions que le code avait une place plus importante que le scénario. Pour changer des jeux narratifs habituels, je me suis dit que ça serait original de le faire sous la forme d'une plateforme de discussion du type Whatsapp, d'autant plus que cela s'intègre bien dans notre époque actuelle, dans laquelle presque tout le monde est équipé d'un téléphone portable.

J'ai également réalisé la fenêtre d'accueil de notre jeu. Il a donc pour cela fallu se renseigner sur la façon dont créer des fenêtres et des widgets avec Tkinter. Naomie et moi avons exploré plusieurs sites et regroupé nos connaissances sur un document drive. Sur cette fenêtre accueil, il y avait aussi un bouton jouer. Lorsque le joueur clique dessus, une autre fenêtre s'ouvre. Pour cela, j'ai relié le bouton jouer à une fonction jouer(). Command=jouer, inscrit au niveau des propriétés du bouton jouer, permet d'appeler cette fonction. Celle-ci cache tout d'abord la fenêtre accueil grâce à ".pack_forget()" puis une autre fenêtre s'ouvre avec une image de messagerie vocale. De plus, dans cette fonction, un enregistrement vocal est chargé.

En effet, j'ai eu l'idée qu'au début du jeu, pour expliquer au joueur le contexte de celui-ci et son objectif, une adolescente dénommée Rose laisserait un message à son ami Paul (que le joueur incarne) en lui demandant de l'aider car un ami à eux, Kevin a disparu. Je voulais alors qu'une fois que le joueur ait cliqué sur le bouton jouer, une image s'affiche (avec écrit messagerie vocale dessus) pendant que le message vocal (enregistré avec Audacity) tourne, puis qu'une fois que celui-ci se termine, une autre fenêtre s'ouvre. Pour cela, j'ai donc commencé à créer une boucle While (tant que) avec un booléen True. J'ai défini un événement contenant une variable INTRO_FINI comme étant vrai (True) et attribué à la variable une valeur quelconque. Puis j'ai créé à l'intérieur des conditions (if et elif). Si l'évènement est réalisé, cela signifie que l'enregistrement est terminé, une autre fenêtre s'ouvre alors. Sinon, le programme continue à chercher toutes les 30 ms si l'événement se réalise. Cette boucle est insérée dans une fonction pour que tout démarre une fois le bouton jouer cliqué. Naomie m'a aussi parfois aidé pour la création de cette boucle. De plus, ce qui a posé des difficultés c'était de trouver comment le programme pouvait détecter le moment exact où l'enregistrement s'arrêtait. En effet, Pycharm démarrait un autre fil d'exécution au moment de l'enregistrement audio. Ainsi, sur le premier fil d'exécution, toute la suite de notre programme continuait à tourner en même temps que l'enregistrement ce que nous ne voulions pas. Pour cela, j'ai finalement trouvé qu'il fallait créer un troisième fil d'exécution (appelé Thread) relié à l'évènement créé ultérieurement (qui ne se réalise qu'au moment où l'enregistrement audio est terminé).

Dans la fenêtre des messages échangés entre le joueur et Rose, on a aussi voulu créer une scrollbar. Je me suis occupé de sa création. Tout d'abord, j'ai défini son orientation (vertical), sa taille et l'ai placé dans un cadre (frame) dans la fenêtre des messages. Mais le problème c'est qu'au fur et à mesure que les messages apparaissaient, la frame avec tous les messages augmentait de taille mais la scrollbar ne s'adaptait pas et on ne pouvait plus la remonter pour voir les messages précédents. J'ai donc dû préciser, à la fin des fonctions, que la frame doive se mettre à jour avec l'arrivée de nouveaux messages pour que la scrollbar puisse se réajuster. J'ai pour cela utilisé la méthode update.idletacks(). Il fallait aussi préciser qu'elle se mette en bas après chaque nouveau message.

Avec Naomie, on s'est réparti le code de l'arbre de choix en deux. J'en ai donc codé la moitié. J'ai ainsi codé les branches qui partent de la réponse "Interroge sa mère". Dans le déroulement du jeu, on avait d'un côté les dialogues qui ne nécessitent pas d'intervention du joueur et qui permettent une mise en situation (incluant les questions et la création des boutons) et de l'autre les réponses à choix que le joueur doit choisir. Il a donc fallu créer deux fonctions : une pour les dialogues "purs" contenant les questions posées au joueur et la création des boutons radios, et une pour les réponses à choix du joueur. Ainsi, lorsque dans la fonction des dialogues appelée affichage_dialogue(), on arrive sur un bouton radio, le programme va chercher, dans un dictionnaire contenant toutes les réponses, celle indiquée (identifiable par son numéro). Puis lorsque le joueur clique sur le bouton "ok" pour valider sa réponse, la fonction traiter_reponse(), qui contient les choix de réponses, est appelée et les conditions de l'arbre à choix selon la réponse sont exécutée. On revient alors à la suite de dialogues et le processus continue déroulant tout l'arbre de choix.

Bilan personnel

Je trouve que le travail en équipe a eu de nombreux avantages. Tout d'abord, étant à deux sur un projet, on a pu se répartir les tâches et ainsi réaliser un projet mieux abouti que si on était seul en un même laps de temps. Travailler avec Naomie m'a aussi apporté de la motivation puisque l'on s'est soutenu mutuellement tout au long de l'aboutissement de notre projet. De plus, cela apporte beaucoup au niveau des connaissances puisqu'un partenaire a souvent des opinions différentes des nôtres ce qui nous fait découvrir de nouveaux points de vue. Cela a été le cas avec Naomie de nombreuses fois comme par exemple pour la réorganisation de notre arbre de choix qui n'était au début pas très clair. Nous n'avions en effet pas la même vision de numérotation et de réorganisation mais le fait de les combiner nous a permis de trouver une méthode simple et efficace de numérotation qui nous a aidé pour nous retrouver dans notre code. Par ailleurs, ce travail d'équipe a été particulier puisque Naomie et moi, habitant tout près, avons pu nous voir très régulièrement pour avancer notre projet ce qui n'est pas le cas pour tout le monde. Cela a alors été très pratique puisque cela nous a permis de nous expliquer des choses en direct ce qui va plus rapidement que par écrit et permet d'avoir une meilleure cohésion d'équipe selon moi.

Amélioration et perspectives

Par manque de temps, nous avons dû mettre de côté certains détails d'améliorations du jeu. De plus, un jeu peut toujours être amélioré.

Tout d'abord, pour que le jeu soit plus attrayant pour le joueur, on aurait pu rajouter la photo de contact de la personne qui envoie le message à côté de chaque message. Pour cela, il aurait fallu créer un canvas dans lequel on aurait mis la photo. Il aurait alors fallu ajouter ce canvas après chaque message (un processus quelque peu long…).

De plus, pour que le jeu soit encore plus interactif, on aurait pu faire en sorte que le joueur puisse écrire ce qu'il veut quand une question lui est posé et que selon ce qu'il répond, le jeu continue à tourner. La réponse du joueur devrait alors être interprété mot par mot. Cela serait alors assez complexe puisque cela ferait sûrement appel à une intelligence artificielle.

Nous restons cependant très contentes et très fières de notre projet. En effet c'était la première fois que nous codions et nous avons réussi à finir notre projet dans les temps (et surtout à le faire fonctionner !), ce qui était notre principale inquiétude. Bien que nous ne pensions pas nous spécialiser dans le domaine de l'informatique plus tard, ce projet nous a ouvert les yeux sur le monde de l'informatique et nous a permis d'apprendre les bases du code.

De plus, nous avons trouvé que même si certains points peuvent paraître simples ou plutôt logiques, il nous a fallu beaucoup de temps pour comprendre les mécanismes nécessaires. C'est à ce moment-là que nous avons réalisé qu'il faut du temps et de la patience pour assimiler des connaissances et pouvoir les appliquer, ce qui n'est pas notre principale qualité… Mais lorsque nous réussissons un objectif nous étions très contente et fière de nous ce qui nous poussait à avancer davantage.

source de l'image de la page d'accueil:
https://www.bing.com/images/search?view=detailV2&ccid=fK1MEYf4&id=6C782F2BE2A321F45E59FC1CE12B918C28140630&thid=OIP.fK1MEYf4vGsetQyNTRnx1AHaHa&mediaurl=https%3a%2f%2fmedia.istockphoto.com%2fvectors%2fsherlock-holmes-detective-vector-id165725124%3fk%3d6%26m%3d165725124%26s%3d612x612%26w%3d0%26h%3dle-uARtMsVc6QfvDco1Ie4YuwwzEmcjf_tpdmapsuxM%3d&exph=612&expw=612&q=image+sherlcok+dessin&simid=608041460192838522&selectedIndex=61&ajaxhist=0

lien github : https://github.com/lasource2018/investigation

Annexes

Premier fichier nommé investigtion.py:

```
1. from tkinter import *
2. import pygame
3. import threading
4. import time
5. import jeu_principal
6. import sys


7. # variable définissant l'événement du morceau sonore d'intro terminé
8. INTRO_FINI = 23


9. # La fonction son_fini nous permet de savoir à quel moment le morceau sonore est terminée grâce à l'événement INTRO_FINI
10.     def son_fini():
11.     while True:
12.     musique_finie = pygame.event.get(INTRO_FINI)
13.     if musique_finie:
14.     # Création de la fenêtre de jeu principal, exécution du jeu
15.     jeu_principal.jeu_principal(fenetre_accueil)
16.     break
17.     else:
18.     time.sleep(0.3)
```

```
19.     # La fonction jouer est exécutée lorsque le bouton jouer
        est cliqué
20.     def jouer():


21.     # cache le cadre accueil
22.     cadre_accueil.pack_forget()
23.     messagerie = PhotoImage(file="messagerie2.png")


24.     canvas_messagerie = Canvas(cadre_msg, width=393,
        height=700)
25.     # Il faut attacher le PhotoImage au widget du Canvas pour
        ne pas que l'image soit perdu http://effbot.org/pyfaq/why-do-
        my-tkinter-images-not-appear.htm
26.     canvas_messagerie.messagerie = messagerie
27.     canvas_messagerie.create_image(0, 0, anchor=NW,
        image=messagerie)
28.     canvas_messagerie.pack()
29.     cadre_msg.pack()


30.     pygame.init()


31.     #chargement du morceau à jouer
32.     pygame.mixer.music.load("messageriePaul.mp3")
33.     pygame.mixer.music.set_endevent(INTRO_FINI)
34.     pygame.mixer.music.play()


35.     # Création d'un autre fil (thread) d'exécution pour
        attraper l'événement de fin de morceau sonore
36.     t = threading.Thread(target=son_fini)
37.     t.start()
```

```
38.     #début du corps principal du code


39.     # création de la fenêtre principale
40.     fenetre_accueil = Tk()
41.     fenetre_accueil.configure(background="white")
42.     fenetre_accueil.wm_minsize(600, 700)
43.     fenetre_accueil.wm_maxsize(600, 700)
44.     fenetre_accueil.wm_title("Menu")


45.     # création d'un cadre accueil
46.     cadre_accueil=Frame(fenetre_accueil, bg="white")
47.     cadre_accueil.pack()


48.     # cadre pour la messagerie
49.     cadre_msg=Frame(fenetre_accueil)


50.     #titre du jeu sur la page d'accueil
51.     titre_label = Label(cadre_accueil, text="Lookin' for
  Kevin", font=("Broadway", 35), bg="white")
52.     titre_label.pack()


53.     #insertion de la photo de la page d'accueil
54.     photo = PhotoImage(file="detective.png")
55.     canvas_photo = Canvas(cadre_accueil, width=550,
  height=550, bg="white")
56.     canvas_photo.create_image(0, 5, anchor=NW, image=photo)
57.     canvas_photo.pack()


58.     #création du bouton jouer
```

```
59.     jouer_bouton = Button(cadre_accueil, width=20, height=5,
        text="JOUER", font=("calibri", 20), bg="light slate blue",
        command=jouer)
60.     jouer_bouton.pack()


61.     fenetre_accueil.mainloop()
```

Deuxième fichier nommé jeu_principal.py:

```
1. from tkinter import *
2. import time
3. import sys


4. # définition de variables récurentes dans le code
5. width=425
6. height=650
7. wraplength=215


8. #définition de l'intervalle de temps entre l'affichage de
   chaque message (en ms)
9. intervalle_temps = 2300


10. #None permet de créer des variables sans aucune valeur
    pour pouvoir les utiliser plus tard
11. fenetre_principal = None
12. fenetre_demarrage = None
13. fr=None
14. message_canvas=None
15. reponse_canvas=None
16. reponse_enigme=None
17. ligne_message=7


18. message_a_afficher = 1


19. #création d'un dictionnaire contenant toutes les réponses
    aux questions à choix
20. liste_reponse={ 'r1' : 'Interroge sa mère pour savoir où
    il est allé en premier',
21. 'r2' : 'Tente de trouver des indices dans sa chambre',
22. 'r6' : 'Celui qui dit que Kevin est allé chez Pierre, un
    ami à lui dans une rue à droite',
```

```
'r7' : 'Celui qui dit que Kevin est allé voir un de ses partenaires de travail dans une rue à gauche',
'r10' : 'où Kevin est allé après être allé le voir',
'r11' : 's\'il est au courant de quelque chose de louche qui pourrait être liée à la disparition de Kevin',
'r12' : 'c\'est louche, on retourne dans la rue de droite',
'r13' : 'on le suit',
'r4' : 'Va chez son oncle',
'r5' : 'Va chez Melina, c'est à 2 min d\'ici',
'r8': 'Bon bah tant pis pars d\'ici on va essayer de trouver des indices ailleurs',
'r9' : 'ça veut dire qu\'elle est à l\'intérieur, enfonce la porte, le temps presse',
'r14' : 'Mme Peletier',
'r15' : 'Camilia',
'r16' : 'Essaie de lui faire cracher le morceau, elle doit couvrir Adam. Dis-lui qu\'on sait que c\'est Calvin et Adam qui ont fait le coup',
'r17' : 'Demande à sa voisine, elle doit savoir quelque chose'}


# variable stockant les réponses des boutons radios
reponse_glob= ""


#création d'un bouton rejouer pour recommencer le jeu
def rejouer():
global message_a_afficher, fr, reponse_canvas, fenetre_principal
message_a_afficher = 1
supprimer_widget(fr)
supprimer_widget(reponse_canvas)
fenetre_principal.after(250, affichage_dialogue)


#création d'un bouton quitter pour quitter le jeu sans le recommencer
def quitter():
sys.exit(0)


#création d'une fonction qui transforme une variable locale en variable globale
def selectionner_reponse(rep):
global reponse_glob
reponse_glob = rep
```

```
#destruction d'un widget
def supprimer_widget(widget_parent):
for widget in widget_parent.grid_slaves():
widget.destroy()


#création de la fonction pour répondre à l'énigme
def traiter_reponse_enigme():
global reponse_enigme, ligne_message, reponse_canvas,
fenetre_principal, message_a_afficher, message_canvas
# transforme tout ce que le joueur écrit en minuscule
reponse = reponse_enigme.get().lower()
# supprime les espaces avant et après un mot
reponse = reponse.strip()


if reponse == "espoir" or reponse == "espérance" or
reponse == "esperance" or reponse == "l'espoir" or
reponse == "l'espérance" or reponse == "l'esperance":
#Enigme résolu
#Label(fr, text=liste_reponse["enigme_resolu"],
borderwidth=2, relief="ridge", bg="white",
wraplength=wraplength).grid(row=ligne_message, column=0,
sticky=W)
#supprimer_widget(reponse_canvas)
message_a_afficher = 16
fenetre_principal.after(200, affichage_dialogue)


else:
message_a_afficher = 200
fenetre_principal.after(200, affichage_dialogue)
#Label(fr, text=liste_reponse["enigme_non_resolu"],
borderwidth=2, relief="ridge", bg="white",
wraplength=wraplength).grid(row=ligne_message, column=0,
sticky=W)


# Mise à jour de la frame
fr.update_idletasks()
message_canvas.config(scrollregion=message_canvas.bbox("a
ll"))
message_canvas.yview_moveto(1)


# traitement des réponses aux questions à choix
def traiter_reponse():
global reponse_glob, ligne_message, message_canvas, fr,
reponse_enigme, reponse_canvas, message_a_afficher
ligne_message += 1
```

```
Label(fr, text=liste_reponse[reponse_glob], borderwidth=2, relief="ridge", bg="pale green", wraplength=wraplength).grid(row=ligne_message, column=1, sticky=E)
# Mise à jour des paramètres du canvas et du frame pour que le scrollbar s'affiche et fasse défiler les réponses
message_canvas.create_window(0, 0, window=fr)
fr.update_idletasks()
message_canvas.config(scrollregion=message_canvas.bbox("all"))
if  (reponse_glob == "r1"):
message_a_afficher = 10
#lancement de l'intervalle de temps et affcihage de la réponse choisie par le joueur
fenetre_principal.after(intervalle_temps, affichage_dialogue)


elif (reponse_glob == "r2"):
message_a_afficher = 100
# efface le canvas des réponses
supprimer_widget(reponse_canvas)
fenetre_principal.after(intervalle_temps, affichage_dialogue)


elif (reponse_glob == "r6"):
message_a_afficher = 20
supprimer_widget(reponse_canvas)
fenetre_principal.after(intervalle_temps, affichage_dialogue)


elif (reponse_glob == "r7"):
    message_a_afficher = 22
    supprimer_widget(reponse_canvas)
    fenetre_principal.after(intervalle_temps, affichage_dialogue)


    elif (reponse_glob == "r10"):
    message_a_afficher = 21.5
    supprimer_widget(reponse_canvas)
    fenetre_principal.after(intervalle_temps, affichage_dialogue)


    elif (reponse_glob == "r11"):
    message_a_afficher = 21.5
```

```
109.     fenetre_principal.after(intervalle_temps,
   affichage_dialogue)

110.     elif (reponse_glob == "r12"):
111.     message_a_afficher = 20
112.     fenetre_principal.after(intervalle_temps,
   affichage_dialogue)

113.     elif (reponse_glob == "r13"):
114.     message_a_afficher = 24
115.     fenetre_principal.after(intervalle_temps,
   affichage_dialogue)

116.     elif (reponse_glob == "r4"):
117.     message_a_afficher = 14
118.     fenetre_principal.after(intervalle_temps,
   affichage_dialogue)

119.     elif (reponse_glob == "r5"):
120.     message_a_afficher = 104
121.     fenetre_principal.after(intervalle_temps,
   affichage_dialogue)

122.     elif (reponse_glob == "r8"):
123.     message_a_afficher = 106
124.     supprimer_widget(reponse_canvas)
125.     fenetre_principal.after(intervalle_temps,
   affichage_dialogue)

126.     elif (reponse_glob == "r9"):
127.     message_a_afficher = 118
128.     supprimer_widget(reponse_canvas)
129.     fenetre_principal.after(intervalle_temps,
   affichage_dialogue)

130.     elif (reponse_glob == "r14"):
131.     message_a_afficher = 110
132.     fenetre_principal.after(intervalle_temps,
   affichage_dialogue)

133.     elif (reponse_glob == "r15"):
134.     message_a_afficher = 114
```

```
135.     fenetre_principal.after(intervalle_temps,
  affichage_dialogue)


136.     elif (reponse_glob == 'r16'):
137.     message_a_afficher = 115
138.     fenetre_principal.after(intervalle_temps,
  affichage_dialogue)


139.     elif (reponse_glob == 'r17'):
140.     message_a_afficher = 122
141.     fenetre_principal.after(intervalle_temps,
  affichage_dialogue)


142.     # suite des messages du dialogue et des questions
143.     def affichage_dialogue():
144.     global message_a_afficher, message_canvas,
  reponse_canvas, fr, ligne_message, reponse_enigme
145.     # Premiers messages
146.     if message_a_afficher == 1:
147.     # wraplenght permet de garder toujours la même
  largeur de canvas maximum (définie auparavant) quelle que
  soit la taille du texte
148.     Label(fr, text="Je vais t'aider Rose", borderwidth=2,
  relief="ridge", bg="pale green",
  wraplength=wraplength).grid(row=0, column=1, sticky=E)
149.     fenetre_principal.after(intervalle_temps,
  affichage_dialogue)
150.     message_a_afficher += 1
151.     elif message_a_afficher == 2:
152.     Label(fr, text="Parfait. Tu es prêt à rester
  connecter avec moi 24h sur 24? ", borderwidth=2,
  relief="ridge", bg="white",
  wraplength=wraplength).grid(row=1, column=0, sticky=W)
153.     fenetre_principal.after(intervalle_temps,
  affichage_dialogue)
154.     message_a_afficher += 1
155.     elif message_a_afficher == 3:
156.     Label(fr, text="Je n'ai pas vraiment le choix de
  toute façon non ?", borderwidth=2, relief="ridge",
  bg="pale green", wraplength=wraplength).grid(row=2,
  column=1, sticky=E)
157.     fenetre_principal.after(intervalle_temps,
  affichage_dialogue)
158.     message_a_afficher += 1
159.     elif message_a_afficher == 4:
```

```
160.     Label(fr, text="haha oui... X-D je pensais commencer
         par chez-lui ", borderwidth=2, relief="ridge", bg="white",
         wraplength=wraplength).grid(row=3, column=0, sticky=W)
161.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
162.     message_a_afficher += 1
163.     elif message_a_afficher == 5:
164.     Label(fr, text="ok mais euh tu vas te pointer chez
         ses parents comme ça : \"bonjour je pense que la vie de
         votre fils est en danger, auriez-vous des renseignements
         pour que je le retrouve ?\"", borderwidth=2,
         relief="ridge", bg="pale green",
         wraplength=wraplength).grid(row=4, column=1, sticky=E)
165.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
166.     message_a_afficher += 1
167.     elif message_a_afficher == 6:
168.     Label(fr, text="Mais non imbécile j'inventerai une
         excuse bidon genre j'ai oublié un devoir chez lui!",
         borderwidth=2, relief="ridge", bg="white",
         wraplength=wraplength).grid(row=5, column=0, sticky=W)
169.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
170.     message_a_afficher += 1
171.     elif message_a_afficher == 7:
172.     Label(fr, text="Ah oui pas bête!", borderwidth=2,
         relief="ridge", bg="pale green").grid(row=6, column=1,
         sticky=E)
173.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
174.     message_a_afficher += 1
175.     elif message_a_afficher == 8:
176.     Label(fr, text="Je suis arrivée chez lui. Je commence
         par quoi?", borderwidth=2, relief="ridge", bg="white",
         wraplength=wraplength).grid(row=7, column=0, sticky=W)
177.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
178.     message_a_afficher += 1
179.     elif message_a_afficher == 9:
180.     reponse = StringVar()
181.     radio1 = Radiobutton(reponse_canvas,
         text=liste_reponse["r1"], variable=reponse, value="r1",
         command=lambda: selectionner_reponse("r1"),
         wraplength=wraplength)
182.     radio2 = Radiobutton(reponse_canvas,
         text=liste_reponse["r2"], variable=reponse, value="r2",
         command=lambda: selectionner_reponse("r2"),
         wraplength=wraplength)
183.     radio1.select()
```

```
184.     selectionner_reponse("r1")
185.     radio2.deselect()
186.     radio1.grid(row=0, column=0)
187.     radio2.grid(row=0, column=1)
188.     okbutton = Button(reponse_canvas, text="OK",
         command=traiter_reponse)
189.     # columnspan est un paramètre qui permet d'étendre le
         widget à plusieurs colonnes
190.     okbutton.grid(row=1, column=0, columnspan=2)
191.     elif message_a_afficher == 10:
192.     ligne_message += 1
193.     Label(fr, text="Bon elle n'est pas très bavarde mais
         elle m'a quand même appris des trucs. Selon elle, Kevin
         est parti voir Amélie", borderwidth=2, relief="ridge",
         bg="white",  wraplength=wraplength).grid(row=ligne_message,
         column=0, sticky=W)
194.     message_a_afficher += 1
195.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
196.     elif message_a_afficher == 11:
197.     ligne_message += 1
198.     Label(fr, text="Ok bon bah va la voir, elle traine
         toujours au café en face de chez lui.", borderwidth=2,
         relief="ridge", bg="pale green",
         wraplength=wraplength).grid(row=ligne_message, column=1,
         sticky=E)
199.     message_a_afficher += 1
200.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
201.     elif message_a_afficher == 12:
202.     ligne_message += 1
203.     Label(fr, text="Oui je viens de la croiser, elle m'a
         dit  qu'elle et Kevin devait déjeuner hier ensemble vers
         12h mais qu'il est parti voir son oncle sans raison donc
         je me dirige vers chez lui", borderwidth=2,
         relief="ridge", bg="white",
         wraplength=wraplength).grid(row=ligne_message, column=0,
         sticky=W)
204.     message_a_afficher += 1
205.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
206.     elif message_a_afficher == 13:
207.     ligne_message += 1
208.     Label(fr, text="ok", borderwidth=2, relief="ridge",
         bg="pale green",
         wraplength=wraplength).grid(row=ligne_message, column=1,
         sticky=E)
209.     message_a_afficher += 1
```

```
210.     fenetre_principal.after(intervalle_temps,
    affichage_dialogue)
211.     elif message_a_afficher == 14:
212.     ligne_message += 1
213.     Label(fr, text="Son oncle dit qu'il veut bien nous
    aider à condition qu'on réponde à une énigme",
    borderwidth=2, relief="ridge", bg="white",
    wraplength=wraplength).grid(row=ligne_message, column=0,
    sticky=W)
214.     message_a_afficher += 1
215.     fenetre_principal.after(intervalle_temps,
    affichage_dialogue)
216.     elif message_a_afficher == 15:
217.     # Enigme de l'oncle
218.     ligne_message += 1
219.     Label(fr, text="Voici l'énigme de l'oncle :\"L'échec
    ne l'arrête pas, il va de pair avec la foi, on dit qu'il
    fait vivre\"", borderwidth=2, relief="ridge", bg="white",
    wraplength=wraplength).grid(row=ligne_message, column=0,
    sticky=W)
220.     # efface le canvas des réponses
221.     supprimer_widget(reponse_canvas)
222.     # mettre le champ de texte pour réponse
223.     reponse_enigme = Entry(reponse_canvas)
224.     reponse_enigme.grid(row=0, column=0, columnspan=2)
225.     Button(reponse_canvas, text="OK",
    command=traiter_reponse_enigme).grid(row=1, column=0,
    columnspan=2)
226.     elif message_a_afficher == 16:
227.     ligne_message +=1
228.     Label(fr, text="Bien joué, c'est ça ! Son oncle m'a
    expliqué qu'il était paniqué quand il est passé le voir
    car il avait reçu un SMS étrange qui disait « nous te
    surveillons Kévin ». Il est ensuite allé rue Martini, j'y
    vais.", borderwidth=2, relief="ridge", bg="white",
    wraplength=wraplength).grid(row=ligne_message, column=0,
    sticky=W)
229.     supprimer_widget(reponse_canvas)
230.     message_a_afficher +=1
231.     fenetre_principal.after(intervalle_temps,
    affichage_dialogue)
232.     elif message_a_afficher == 200:
233.     ligne_message += 1
234.     Label(fr, text="L\'oncle dit que ce n\'est pas cela,
    essaie autre chose.", borderwidth=2, relief="ridge",
    bg="white",  wraplength=wraplength).grid(row=ligne_message,
    column=0, sticky=W)
235.     elif message_a_afficher == 17:
236.     ligne_message += 1
```

```
237.     Label(fr, text="Tu vois quelqu'un ?", borderwidth=2,
   relief="ridge", bg="pale green",
   wraplength=wraplength).grid(row=ligne_message, column=1,
   sticky=E)
238.     message_a_afficher +=1
239.     fenetre_principal.after(intervalle_temps,
   affichage_dialogue)
240.     elif message_a_afficher == 18:
241.     ligne_message += 1
242.     Label(fr, text="Oui deux commerçants mais qui ont une
   version différente. Lequel j'écoute ?", borderwidth=2,
   relief="ridge", bg="white",
   wraplength=wraplength).grid(row=ligne_message, column=0,
   sticky=W)
243.     message_a_afficher +=1
244.     fenetre_principal.after(intervalle_temps,
   affichage_dialogue)
245.     elif message_a_afficher == 19:
246.     reponse = StringVar()
247.     radio1 = Radiobutton(reponse_canvas,
   text=liste_reponse["r6"], variable=reponse, value="r1",
   command=lambda: selectionner_reponse("r6"),
   wraplength=wraplength)
248.     radio2 = Radiobutton(reponse_canvas,
   text=liste_reponse["r7"], variable=reponse, value="r2",
   command=lambda: selectionner_reponse("r7"),
   wraplength=wraplength)
249.     radio1.select()
250.     selectionner_reponse("r6")
251.     radio2.deselect()
252.     radio1.grid(row=0, column=0)
253.     radio2.grid(row=0, column=1)
254.     okbutton = Button(reponse_canvas, text="OK",
   command=traiter_reponse)
255.     # columnspan est un paramètre qui permet d'étendre le
   widget à plusieurs colonnes
256.     okbutton.grid(row=1, column=0, columnspan=2)
257.     elif message_a_afficher == 20:
258.     ligne_message += 1
259.     Label(fr, text="Je suis chez Pierre, je lui demande
   quoi?", borderwidth=2, relief="ridge", bg="white",
   wraplength=wraplength).grid(row=ligne_message, column=0,
   sticky=W)
260.     message_a_afficher += 1
261.     fenetre_principal.after(intervalle_temps,
   affichage_dialogue)
262.     elif message_a_afficher == 21:
263.     reponse = StringVar()
```

```
264.     radio1 = Radiobutton(reponse_canvas,
   text=liste_reponse["r10"], variable=reponse, value="r1",
   command=lambda: selectionner_reponse("r10"),
   wraplength=wraplength)
265.     radio2 = Radiobutton(reponse_canvas,
   text=liste_reponse["r11"], variable=reponse, value="r2",
   command=lambda: selectionner_reponse("r11"),
   wraplength=wraplength)
266.     radio1.select()
267.     selectionner_reponse("r10")
268.     radio2.deselect()
269.     radio1.grid(row=0, column=0)
270.     radio2.grid(row=0, column=1)
271.     okbutton = Button(reponse_canvas, text="OK",
   command=traiter_reponse)
272.     # columnspan est un paramètre qui permet d'étendre le
   widget à plusieurs colonnes
273.     okbutton.grid(row=1, column=0, columnspan=2)
274.     elif message_a_afficher == 21.5:
275.     ligne_message += 1
276.     Label(fr, text="Pierre m'a dit qu'après être allé
   chez lui, Kevin est allé voir Melina", borderwidth=2,
   relief="ridge", bg="white",
   wraplength=wraplength).grid(row=ligne_message, column=0,
   sticky=W)
277.     supprimer_widget(reponse_canvas)
278.     message_a_afficher = 104
279.     fenetre_principal.after(intervalle_temps,
   affichage_dialogue)
280.     elif message_a_afficher == 22:
281.     ligne_message += 1
282.     Label(fr, text="J'ai croisé un homme dans la rue, il
   dit qu'il sait où est Kevin. Je fais quoi?",
   borderwidth=2, relief="ridge", bg="white",
   wraplength=wraplength).grid(row=ligne_message, column=0,
   sticky=W)
283.     supprimer_widget(reponse_canvas)
284.     message_a_afficher += 1
285.     fenetre_principal.after(intervalle_temps,
   affichage_dialogue)
286.     elif message_a_afficher == 23:
287.     reponse = StringVar()
288.     radio1 = Radiobutton(reponse_canvas,
   text=liste_reponse["r12"], variable=reponse, value="r1",
   command=lambda: selectionner_reponse("r12"),
   wraplength=wraplength)
289.     radio2 = Radiobutton(reponse_canvas,
   text=liste_reponse["r13"], variable=reponse, value="r2",
```

```
    command=lambda: selectionner_reponse("r13"),
    wraplength=wraplength)
290.     radio1.select()
291.     selectionner_reponse("r12")
292.     radio2.deselect()
293.     radio1.grid(row=0, column=0)
294.     radio2.grid(row=0, column=1)
295.     okbutton = Button(reponse_canvas, text="OK",
    command=traiter_reponse)
296.     okbutton.grid(row=1, column=0, columnspan=2)
297.     elif message_a_afficher == 24:
298.     # PARTIE PERDUE
299.     ligne_message += 1
300.     Label(fr, text="L'homme m'a conduit à un garçon nommé
    Kevin. Mais ça n'est pas notre Kevin.", borderwidth=2,
    relief="ridge", bg="white",
    wraplength=wraplength).grid(row=ligne_message, column=0,
    sticky=W)
301.     message_a_afficher += 1
302.     fenetre_principal.after(intervalle_temps,
    affichage_dialogue)
303.     elif message_a_afficher == 25:
304.     # PARTIE PERDUE
305.     ligne_message += 1
306.     Label(fr, text="Oh non...j'y ai tellement cru...",
    borderwidth=2, relief="ridge", bg="pale green",
    wraplength=wraplength).grid(row=ligne_message, column=1,
    sticky=E)
307.     message_a_afficher += 1
308.     fenetre_principal.after(intervalle_temps,
    affichage_dialogue)
309.     elif message_a_afficher == 26:
310.     # PARTIE PERDUE
311.     ligne_message += 1
312.     Label(fr, text="Oh non Paul, mon dieu...",
    borderwidth=2, relief="ridge", bg="white",
    wraplength=wraplength).grid(row=ligne_message, column=0,
    sticky=W)
313.     message_a_afficher += 1
314.     fenetre_principal.after(intervalle_temps,
    affichage_dialogue)
315.     elif message_a_afficher == 27:
316.     # PARTIE PERDUE
317.     ligne_message += 1
318.     Label(fr, text="Quoi? Qu'est-ce qu'il y a?",
    borderwidth=2, relief="ridge", bg="pale green",
    wraplength=wraplength).grid(row=ligne_message, column=1,
    sticky=E)
```

```
319.     message_a_afficher += 1
320.     fenetre_principal.after(intervalle_temps,
   affichage_dialogue)
321.     elif message_a_afficher == 28:
322.     # PARTIE PERDUE
323.     ligne_message += 1
324.     Label(fr, text="J'ai reçu un appel, le corps de Kevin
   a été retrouvé par un passant, une balle dans la
   poitrine.", borderwidth=2, relief="ridge", bg="white",
   wraplength=wraplength).grid(row=ligne_message, column=0,
   sticky=W)
325.     supprimer_widget(reponse_canvas)
326.     Button(reponse_canvas, text="Rejouer",
   command=rejouer).grid(row=1, column=0)
327.     Button(reponse_canvas, text="Quitter",
   command=quitter).grid(row=1, column=1)
328.     elif message_a_afficher == 100:
329.     ligne_message += 1
330.     Label(fr, text="J'ai trouvé un post-it sur lequel il
   y a marqué \"samedi 14h rdv avec Pierre\" il habite pas
   loin j'y vais ", borderwidth=2, relief="ridge",
   bg="white",  wraplength=wraplength).grid(row=ligne_message,
   column=0, sticky=W)
331.     message_a_afficher += 1
332.     fenetre_principal.after(intervalle_temps,
   affichage_dialogue)
333.     elif message_a_afficher == 101:
334.     ligne_message += 1
335.     Label(fr, text="Ok demande lui s'il est au courant
   d'un truc qui aurait un lien avec la disparition de
   Kevin", borderwidth=2, relief="ridge", bg="pale green",
   wraplength=wraplength).grid(row=ligne_message, column=1,
   sticky=E)
336.     message_a_afficher += 1
337.     fenetre_principal.after(intervalle_temps,
   affichage_dialogue)
338.     elif message_a_afficher == 102:
339.     ligne_message += 1
340.     Label(fr, text="Je viens de le faire . Il m'a dit
   qu'il avait entendu quelqu'un dire \"si dimanche 18h30,
   Kevin ne nous l'a pas donné, on l'élimine\" Par contre il
   ne sait rien d'autre et le temps presse. Je vais où ?",
   borderwidth=2, relief="ridge", bg="white",
   wraplength=wraplength).grid(row=ligne_message, column=0,
   sticky=W)
341.     message_a_afficher += 1
342.     fenetre_principal.after(intervalle_temps,
   affichage_dialogue)
343.     elif message_a_afficher == 103:
```

```
344.     reponse = StringVar()
345.     radio1 = Radiobutton(reponse_canvas,
         text=liste_reponse["r4"], variable=reponse, value="r1",
         command=lambda: selectionner_reponse("r4"),
         wraplength=wraplength)
346.     radio2 = Radiobutton(reponse_canvas,
         text=liste_reponse["r5"], variable=reponse, value="r2",
         command=lambda: selectionner_reponse("r5"),
         wraplength=wraplength)
347.     radio1.select()
348.     selectionner_reponse("r4")
349.     radio2.deselect()
350.     radio1.grid(row=0, column=0)
351.     radio2.grid(row=0, column=1)
352.     okbutton = Button(reponse_canvas, text="OK",
         command=traiter_reponse)
353.     # columnspan est un paramètre qui permet d'étendre le
         widget à plusieurs colonnes, ici on veut le centrer
354.     okbutton.grid(row=1, column=0, columnspan=2)
355.     elif message_a_afficher == 104:
356.     ligne_message +=1
357.     Label(fr, text="ça y est je suis devant chez Melina,
         mais elle ne répond pas. Y a de la lumière à l'intérieur,
         je fais quoi ?", borderwidth=2, relief="ridge",
         bg="white",  wraplength=wraplength).grid(row=ligne_message,
         column=0, sticky=W)
358.     message_a_afficher += 1
359.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
360.     elif message_a_afficher == 105:
361.     reponse = StringVar()
362.     radio1 = Radiobutton(reponse_canvas,
         text=liste_reponse["r8"], variable=reponse, value="r1",
         command=lambda: selectionner_reponse("r8"),
         wraplength=wraplength)
363.     radio2 = Radiobutton(reponse_canvas,
         text=liste_reponse["r9"], variable=reponse, value="r2",
         command=lambda: selectionner_reponse("r9"),
         wraplength=wraplength)
364.     radio1.select()
365.     selectionner_reponse("r8")
366.     radio2.deselect()
367.     radio1.grid(row=0, column=0)
368.     radio2.grid(row=0, column=1)
369.     okbutton = Button(reponse_canvas, text="OK",
         command=traiter_reponse)
370.     # columnspan est un paramètre qui permet d'étendre le
         widget à plusieurs colonnes
371.     okbutton.grid(row=1, column=0, columnspan=2)
```

```
372.     elif message_a_afficher== 106:
373.     ligne_message += 1
374.     Label(fr, text="Ok je passe dans la rue derrière,
         purée, je vois un truc, on dirait le gant que j'ai offert
         à Kevin. Y a deux personnes dans la rue, je vais les
         interroger", borderwidth=2, relief="ridge", bg="white",
         wraplength=wraplength ).grid(row=ligne_message, column=0,
         sticky=W)
375.     message_a_afficher += 1
376.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
377.     elif message_a_afficher== 107:
378.     ligne_message += 1
379.     Label(fr, text="Et alors ?", borderwidth=2,
         relief="ridge", bg="pale green",
         wraplength=wraplength).grid(row=ligne_message, column=1,
         sticky=E)
380.     message_a_afficher += 1
381.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
382.     elif message_a_afficher== 108:
383.     ligne_message += 1
384.     Label(fr, text="Mme Peletier dit qu'elle a aperçu un
         jeune Homme poursuivi par deux autres personnes et
         Camilia affirme qu'elle a vu Kevin marchait seul. A ton
         avis, laquelle ment ?", borderwidth=2, relief="ridge",
         bg="white",  wraplength=wraplength).grid(row=ligne_message,
         column=0, sticky=W)
385.     message_a_afficher += 1
386.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
387.     elif message_a_afficher== 109:
388.     reponse = StringVar()
389.     radio1 = Radiobutton(reponse_canvas,
         text=liste_reponse["r14"], variable=reponse, value="r1",
         command=lambda: selectionner_reponse("r14"),
         wraplength=wraplength)
390.     radio2 = Radiobutton(reponse_canvas,
         text=liste_reponse["r15"], variable=reponse, value="r2",
         command=lambda: selectionner_reponse("r15"),
         wraplength=wraplength)
391.     radio1.select()
392.     selectionner_reponse("r14")
393.     radio2.deselect()
394.     radio1.grid(row=0, column=0)
395.     radio2.grid(row=0, column=1)
396.     okbutton = Button(reponse_canvas, text="OK",
         command=traiter_reponse)
```

```
397.     # columnspan est un paramètre qui permet d'étendre le
         widget à plusieurs colonnes, ici on veut le centrer
398.     okbutton.grid(row=1, column=0, columnspan=2)
399.     elif message_a_afficher== 110:
400.     ligne_message += 1
401.     Label(fr, text="Ok j'ai interrogé Mme Peletier et
         elle a dit la même chose", borderwidth=2, relief="ridge",
         bg="white",  wraplength=wraplength).grid(row=ligne_message,
         column=0, sticky=W)
402.     message_a_afficher += 1
403.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
404.     elif message_a_afficher == 111:
405.     # PARTIE PERDUE
406.     ligne_message += 1
407.     Label(fr, text="Oh non Paul, mon dieu...",
         borderwidth=2, relief="ridge", bg="white",
         wraplength=wraplength).grid(row=ligne_message, column=0,
         sticky=W)
408.     message_a_afficher += 1
409.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
410.     elif message_a_afficher == 112:
411.     # PARTIE PERDUE
412.     ligne_message += 1
413.     Label(fr, text="Quoi? Qu'est-ce qu'il y a?",
         borderwidth=2, relief="ridge", bg="pale green",
         wraplength=wraplength).grid(row=ligne_message, column=1,
         sticky=E)
414.     message_a_afficher += 1
415.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
416.     elif message_a_afficher == 113:
417.     # PARTIE PERDUE
418.     ligne_message += 1
419.     Label(fr, text="J'ai reçu un appel, le corps de Kevin
         a été retrouvé par un passant, une balle dans la
         poitrine.", borderwidth=2, relief="ridge", bg="white",
         wraplength=wraplength).grid(row=ligne_message, column=0,
         sticky=W)
420.     supprimer_widget(reponse_canvas)
421.     Button(reponse_canvas, text="Rejouer",
         command=rejouer).grid(row=1, column=0)
422.     Button(reponse_canvas, text="Quitter",
         command=quitter).grid(row=1, column=1)
423.     elif message_a_afficher == 114:
424.     ligne_message += 1
```

```
425.     Label(fr, text="Ok je vais la réinterroger du coups
         pour tenter d'en apprendre plus", borderwidth=2,
         relief="ridge", bg="white",
         wraplength=wraplength).grid(row=ligne_message, column=0,
         sticky=W)
426.     message_a_afficher += 1
427.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
428.     elif message_a_afficher == 115:
429.     ligne_message += 1
430.     Label(fr, text="ça n'a pas été facile mais elle a
         fini par me parler. Elle savait que Calvin et Adam en
         voulait à Kevin et a donc voulu couvrir Adam au cas où il
         se passerait quelque chose. Mais elle n'imaginait pas
         qu'ils pourraient s'en prendre à lui... Quand je lui ai
         parlé, elle a réalisé la gravité de la situation. Elle
         m'a dit que Calvin et Adam avait parlé d'un hangard à 100
         mètres d'ici", borderwidth=2, relief="ridge", bg="white",
         wraplength=wraplength).grid(row=ligne_message, column=0,
         sticky=W)
431.     message_a_afficher += 1
432.     fenetre_principal.after(10000, affichage_dialogue)
433.     elif message_a_afficher == 116:
434.     #partie gagnée
435.     ligne_message += 1
436.     Label(fr, text="super !", borderwidth=2,
         relief="ridge", bg="pale green",
         wraplength=wraplength).grid(row=ligne_message, column=1,
         sticky=E)
437.     message_a_afficher += 1
438.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
439.     elif message_a_afficher == 117:
440.     #partie gagnée
441.     ligne_message += 1
442.     Label(fr, text="Kevin est là Paul!!! On a réussi,
         merci bp pour ton aide !", borderwidth=2, relief="ridge",
         bg="white",  wraplength=wraplength).grid(row=ligne_message,
         column=0, sticky=W)
443.     supprimer_widget(reponse_canvas)
444.     Button(reponse_canvas, text="Rejouer",
         command=rejouer).grid(row=1, column=0)
445.     Button(reponse_canvas, text="Quitter",
         command=quitter).grid(row=1, column=1)


446.     elif message_a_afficher == 118:
447.     ligne_message += 1
```

```
448.     Label(fr, text="J'ai bien fait de rentrer, elle me
         dit que Kevin vendait un produit dopant et qu'il ne
         voulait pas le vendre à Adam et Calvin. ", borderwidth=2,
         relief="ridge", bg="white",
         wraplength=wraplength).grid(row=ligne_message, column=0,
         sticky=W)
449.     message_a_afficher += 1
450.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
451.     elif message_a_afficher == 119:
452.     ligne_message += 1
453.     Label(fr, text="Ok, demande à Camila ce qu'elle sait,
         elle est proche de Adam.", borderwidth=2, relief="ridge",
         bg="pale green",
         wraplength=wraplength).grid(row=ligne_message, column=1,
         sticky=E)
454.     message_a_afficher += 1
455.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
456.     elif message_a_afficher == 120 :
457.     ligne_message += 1
458.     Label(fr, text="Elle dit qu'elle ne sait rien, qu'est
         ce que je fais ?", borderwidth=2, relief="ridge",
         bg="white",  wraplength=wraplength).grid(row=ligne_message,
         column=0, sticky=W)
459.     message_a_afficher += 1
460.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
461.     elif message_a_afficher == 121:
462.     reponse = StringVar()
463.     radio1 = Radiobutton(reponse_canvas,
         text=liste_reponse["r16"], variable=reponse, value="r1",
         command=lambda: selectionner_reponse("r16"),
         wraplength=wraplength)
464.     radio2 = Radiobutton(reponse_canvas,
         text=liste_reponse["r17"], variable=reponse, value="r2",
         command=lambda: selectionner_reponse("r17"),
         wraplength=wraplength)
465.     radio1.select()
466.     selectionner_reponse("r16")
467.     radio2.deselect()
468.     radio1.grid(row=0, column=0)
469.     radio2.grid(row=0, column=1)
470.     okbutton = Button(reponse_canvas, text="OK",
         command=traiter_reponse)
471.     # columnspan est un paramètre qui permet d'étendre le
         widget à plusieurs colonnes, ici on veut le centrer
472.     okbutton.grid(row=1, column=0, columnspan=2)
473.     elif message_a_afficher == 122:
```

```
474.     ligne_message += 1
475.     Label(fr, text="Elle a dit qu'elle avait vu un jeune
         homme se faire poursuivre par deux autres dans sa rue.
         Ils l'ont ensuite attrappé et sont rentrés dans un hangar.
         Elle m'a montré où il était, j'y vais.", borderwidth=2,
         relief="ridge", bg="white",
         wraplength=wraplength).grid(row=ligne_message, column=0,
         sticky=W)
476.     fenetre_principal.after(intervalle_temps,
         affichage_dialogue)
477.     message_a_afficher = 116


478.     # Mise à jour de la frame
479.     fr.update_idletasks()
480.     message_canvas.config(scrollregion=message_canvas.bbo
         x("all"))
481.     message_canvas.yview_moveto(1)


482.     def jeu_principal(fenetre_accueil):
483.     global width, height, fr, reponse_canvas,
         message_canvas, fenetre_principal, fenetre_demarrage


484.     fenetre_principal = Tk()


485.     fenetre_principal.wm_minsize(width, height)
486.     #fenetre_principal.wm_maxsize(500, 600)
487.     fenetre_principal.wm_title("Whats'up")


488.     #Afficher le nom du contact
489.     nomContact = Label(fenetre_principal, text="Rose <3",
         bg="white")
490.     nomContact.grid(row=0, column=0)


491.     #création d'une barre de défilement
492.     barre_defiler = Scrollbar(fenetre_principal,
         orient=VERTICAL)
493.     #création d'un canvas pour mettre la frame des
         message
494.     message_canvas=Canvas(fenetre_principal, width=width,
         height=height - 100)


495.     # Lorsque le contenu dépasse la scrollbar va
         permettre de faire défiler le contenu qui dépasse
496.     barre_defiler.config(command=message_canvas.yview)
```

```
497.     message_canvas.config(yscrollcommand=barre_defiler.se
         t)

498.     message_canvas.grid(row=1, column=0)
499.     barre_defiler.grid(row=1, column=1, sticky=N+S)

500.     fr = Frame(message_canvas, width=width, height=height
         - 100)

501.     fenetre_principal.grid_rowconfigure(1, weight=1)
502.     fenetre_principal.grid_columnconfigure(0, weight=1)

503.     # méthode pour mettre en place le scrollbar
504.     message_canvas.create_window(0, 0, window=fr)
505.     # définition du scroll pour faire défiler les widgets
         dans le canvas, placer pour que les messages soient
         placés au milieu de la frame
506.     message_canvas.configure(scrollregion=message_canvas.
         bbox("all"))

507.     #création d'un cadre pour insérer les choix de
         réponse (un cadre principale, un cadre par serie de choix
         de réponses
508.     reponse_canvas = Canvas(fenetre_principal,
         width=width, height=100)
509.     reponse_canvas.grid(row=2, column=0)

510.     # affichage des messages après 500ms
511.     fenetre_principal.after(250, affichage_dialogue)

512.     fenetre_principal.mainloop()
```