Simon's Adventure

BY SUPERDRY

Sommaire

By Superdry

Pierre Cry et
Matthias Ambroise

Règles du jeu

- Jeu d’aventure
- Dialogues avec personnages
- Système de choix et d’énigmes
- Interaction avec le décor (objets)
- Différentes fins, dépendantes des choix

Structures de données

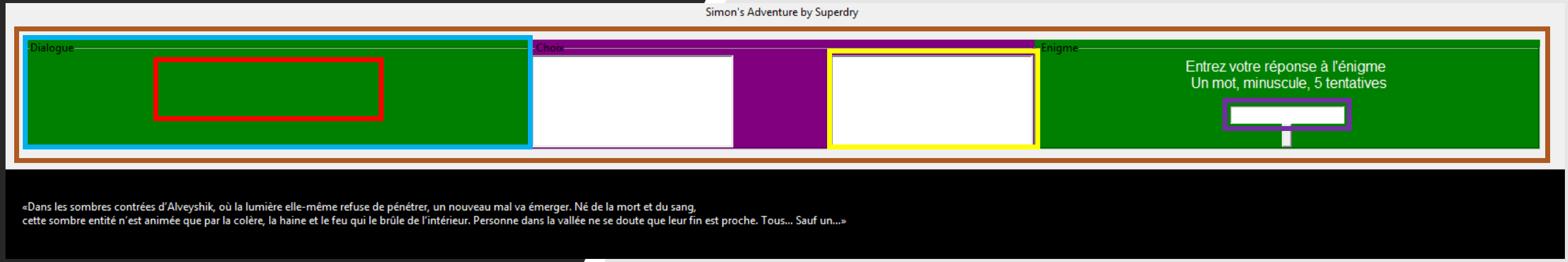
❑ **1 Interface :** ▪ Frame ▪ Label Frame ▪ Label ▪ Entry ▪ Bouton	❑ **38 fonctions :** ▪ 6 fonctions pour le déplacement ▪ 4 fonctions pour le décor et les transitions ▪ 4 fonctions pour les dialogues (dont 1 général) ▪ 8 fonction pour les choix (simples et moraux) ▪ 2 fonctions pour les énigmes ▪ 14 fonctions « secondaires »
❑ **41 variables :** ▪ 6 pour le son ▪ 24 listes (dont 18 listes de texte) ▪ 1 liste de matrices ▪ 6 compteurs ▪ 2 variables de contrôle (fait et faux) ▪ 2 variables de statue (puissance et moral)	❑ **63 images :** ▪ 60 décors ▪ 2 personnages ▪ 1 image de menu ❑ **Son :** ▪ 2 musiques de fond ▪ 5 bruitages

Quelques fonctions - Interface

```
#Frame principale
Interface = Frame(fen, width=1650, height=220, pady=20)
Interface.pack(side=TOP)

#Frame qui contient les dialogues
FrameDiag = LabelFrame(Interface, text="Dialogue", bg="green", width=550, height=120)
FrameDiag.pack(side = LEFT)
FrameDiag.pack_propagate(False)

#Label qui affiche les dialogues
LabelDiag = Label(FrameDiag, text="", fg="white", bg="green")
LabelDiag.pack()

#Frame qui contient les choix
FrameCh = LabelFrame(Interface, text="Choix", bg="purple", width=550, height=120)
FrameCh.pack(side = LEFT)
FrameCh.pack_propagate(False)

#Bouttons utilisés dans le système de choix
ButtonCh1 = Radiobutton(FrameCh, text="", indicatoron=0, width=30, height=30, command=NONE)
ButtonCh1.pack(side=LEFT)
ButtonCh2 = Radiobutton(FrameCh, text="", indicatoron=0, width=30, height=30, command=NONE)
ButtonCh2.pack(side=RIGHT)

#Frame qui contient les enigmes
FrameEnig = LabelFrame(Interface, text="Enigme", bg="green", width=550, height=120)
FrameEnig.pack(side = LEFT)
FrameEnig.pack_propagate(False)

#Label, Bouton et Entry qui gère le système d'enigme
enig = StringVar()
LabelEnig = Label(FrameEnig, text="Entrez votre réponse à l'énigme \n Un mot, minuscule, 5 tentatives", bg="green", fg="white", font="5")
LabelEnig.pack(side=TOP)
ButtonEnig = Button(FrameEnig)
ButtonEnig.pack(side=BOTTOM)
EntryEnig = Entry(FrameEnig, textvariable=enig)
EntryEnig.pack(side=BOTTOM)
```

Quelques fonctions - Interface

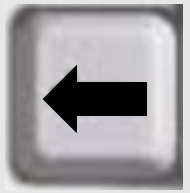
Aperçu

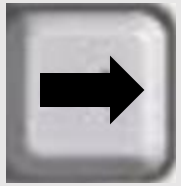


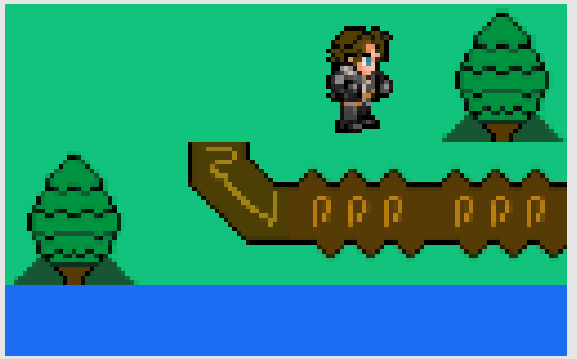
Frame
Label Frame
Label
Bouton
Entry

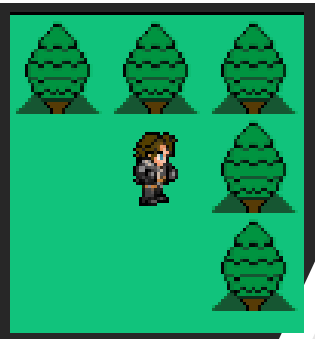
Quelques fonctions – Mouvement et collision

```
##Fonctions de mouvement##
def mouvement(a,b):
    global x, y, img, M, cptM, cptV, cptG
    x, y = x + a, y + b
    if collision(x,y)==True: #Si l'endroit n'est pas un mur, faire le mouvement
        can.coords(img, x, y)
        text("")
        cptM, cptV, cptG, cptchoix=0,0,0,0
    else: #Revenir à la place initiale
        x,y=x-a,y-b

def gauche(event):
    global img, x, y
    if x > 82:
        mouvement(-32, 0)
        can.itemconfigure(img, image=fichier_imgg)

def droite(event):
    global img, x, y
    if x < 1138:
        mouvement(32, 0)
        can.itemconfigure(img, image=fichier_imgd)
```







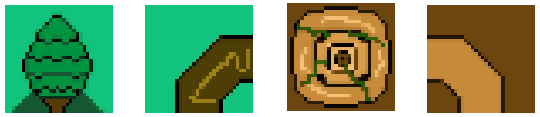


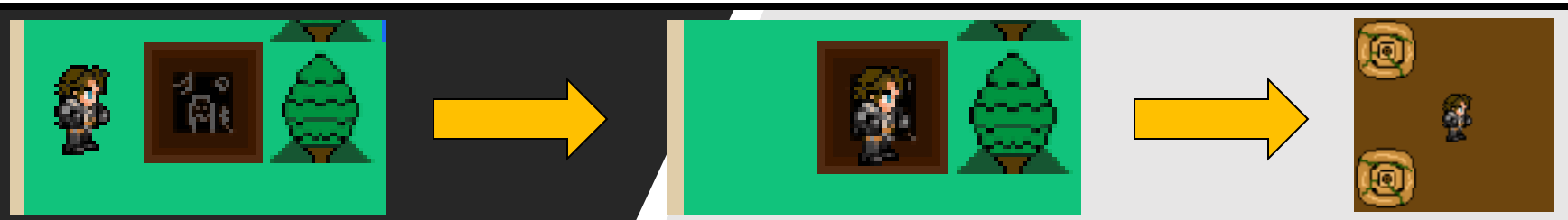
```
##Fonction de collision##
def collision(x,y):
    global M, cptmap
    MUR =[8,9,18,19]
    i,j=indices(x,y)
    if M[cptmap][i][j] not in MUR: #Si l'endroit où souhaite aller le joueur n'est pas un mur
        return True
```

Quelques fonctions – Décor et transition

```
##Fonction qui gère les transitions##
def transition(event):
    global x, y, cptmap
    X1=[242,114,1042]
    Y1=[420,164,164]
    X2=[210,146,1010]
    Y2=[420,164,164]

    if cptmap == -1: #Menu principal
        carte(1010,452,0)
        can.create_text(1400,350,text="Commandes \nDéplacement : "+
                "Touches Directionnelles \nRamassage : z \nChanger de zone : "+
                "a \nInteragir avec un personnage : p \nDéfier la bête : space", fill="white", font=10)
        transit.play()
    for k in range (1,4): #Transition entre les différentes maps
        if cptmap==0 and (x == X1[k-1] and y == Y1[k-1]): #Transition de la map0 aux différentes maps
            carte(86,324,k)
            transit.play()
        if cptmap==k and (x == 54 and y == 324): #Transition des différentes maps à la map0
            carte(X2[k-1],Y2[k-1],0)
            transit.play()
```

```
##Création des décors##
def decor(M):
    global d
    for i in range(len(M)): #Parcourt la liste M
        for j in range(len(M[0])): #Parcourt une matrice de la liste M
            can.create_image(50 + 32 * i, 100 + 32 * j, image=d[M[i][j] - 1])
```

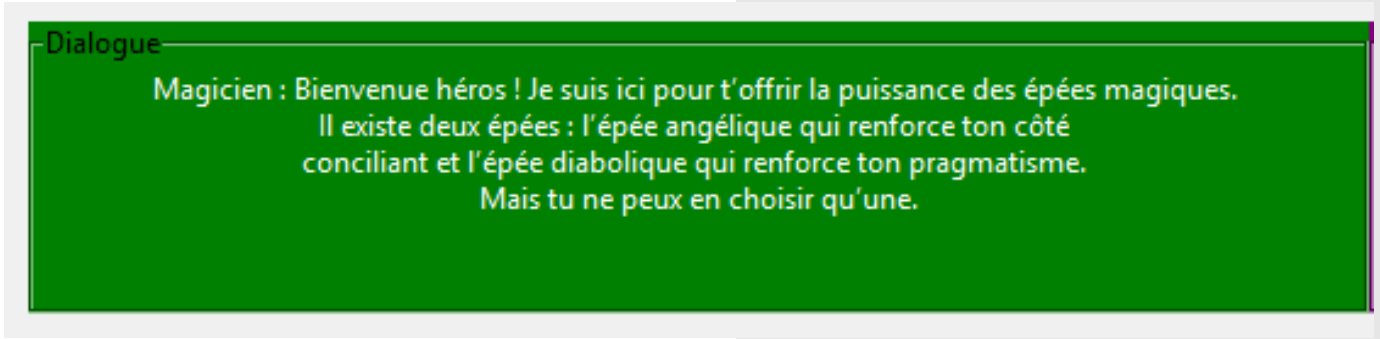




Quelques fonctions – Dialogues

```
#Dialogue#
def Dialogue(event): #Active les dialogues en pressant la touche définit
    DialogueMag()
    DialogueGard()
    DialogueVill()
```

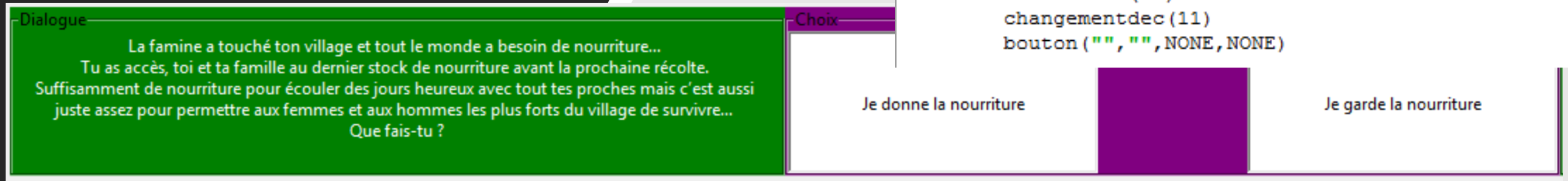
```
def DialogueMag(): #Dialogue avec le magicien
    global M, cptM, fait
    i,j=indices(x,y)
    if M[0][i][j]==48:#Le joueur parle sur la map0 au magicien
        if cptM==0: #Active le son perso
            perso.play()
        if cptM < 2: #Dialogue avant le choix
            text(DiagMag1[cptM])
        if cptM == 2: #Active le choix
            text(DiagMag1[cptM])
            bouton("Oui", "Non",Choix1Mag,Choix2Mag)
        if fait == 1 : #Le joueur a fait le choix
            text(DiagMag1[len(DiagMag1)-1])
            changementdec(10)
            fait=0
    #Le joueur parle pour la deuxième fois au Magicien sur la Map1 après avoir ramassé tous les objets demandé
    if M[1][i][j]==51 and cptM < len(DiagMag3) and élixir()==True:
        if cptM==0: #Active le son perso
            perso.play()
        text(DiagMag3[cptM])
        #Lorsque la liste des dialogues est parcourut, le joueur s'apprête à faire les choix moraux
        if cptM == len(DiagMag3)-1:
            ChoixMor1()
    else:
        #Le joueur parle pour la première fois au Magicien sur la Map1
        if M[1][i][j]==51 and cptM < len(DiagMag2):
            if cptM==0:
                perso.play()
            text(DiagMag2[cptM])

    cptM=cptM+1 #Après que les conditions dépendant de cptM sont executés, le compteur prend +1
```



Quelques fonctions – Choix

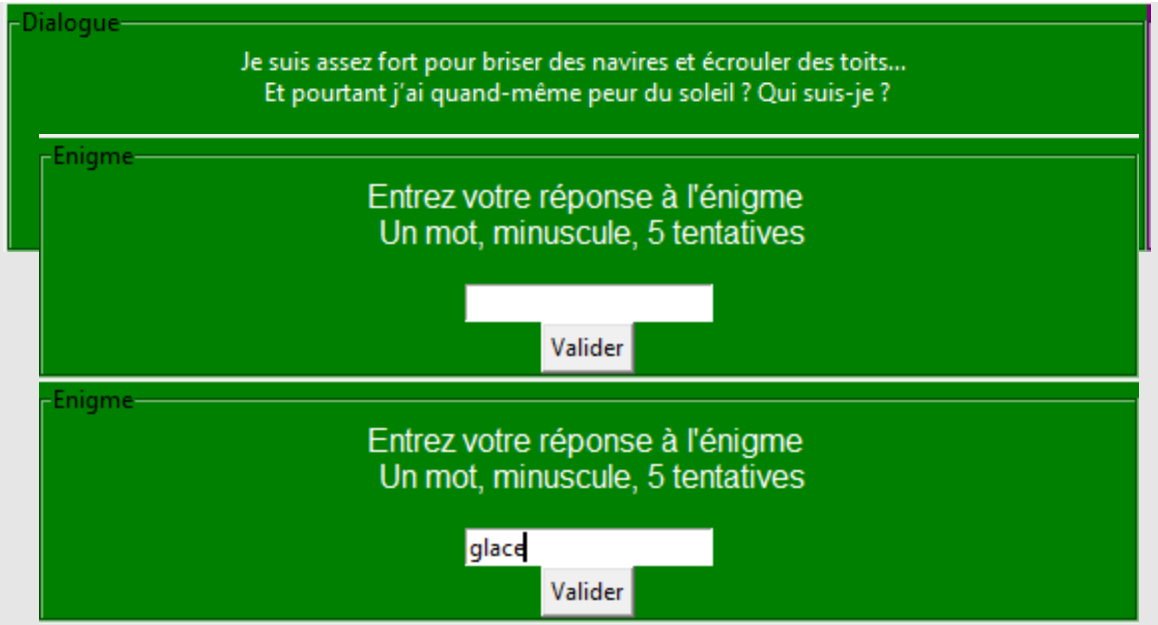
```
def Choix1Gard(): #Le joueur a effectué la première réponse au choix du Garde
    global fait, cptG
    fait=1
    cptG=0
    bouton("","",NONE,NONE)

def Choix2Gard(): #Le joueur a effectué la deuxième réponse au choix du Garde
    global fait, cptG
    fait=-1
    cptG=0
    bouton("","",NONE,NONE)
```


```
#Fonctions qui gère les choix moraux
def ChoixMor1():
    global cptchoix
    for k in range(3):
        if cptchoix==k: #Le joueur effectue le k-ème Choix Moraux
            text(ChMoraux Q[k])
            bouton(ChMoraux RC[k],ChMoraux RP[k],Concilliant,Pragmatique)
    if cptchoix==3: #Le joueur a effectué tous les choix moraux
        MoraxFin()
        changementdec(11)

def Concilliant(): #Concilliant amène à ajouter 1 au moral
    global moral,cptchoix
    moral=moral+1
    cptchoix=cptchoix+1
    ChoixMor1()

def Pragmatique(): #Pragmatique amène à enlever 1 au moral
    global moral,cptchoix
    moral=moral-1
    cptchoix=cptchoix+1
    ChoixMor1()

def MoraxFin(): #Annonce le côté Pragmatique ou Concilliant du joueur
    global moral
    if moral>0: #Joueur Conciliant
        text(ChMoraxFin[0])
        inventaire(38)
        changementdec(11)
        bouton("","",NONE,NONE)
    else: #Joueur Pragmatique
        text(ChMoraxFin[1])
        inventaire(39)
        changementdec(11)
        bouton("","",NONE,NONE)
```



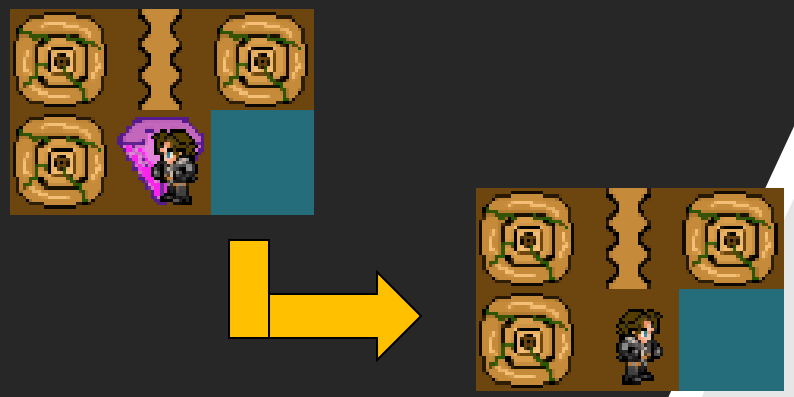
Quelques fonctions – Enigmes

```
#Enigme#
def Enigme():   #Fonction qui assure le déroulement des énigmes
    global cptenig
    ButtonEnig.config(text="Valider", command=verif)
    if cptenig<len(EnigVillQ):
        text(EnigVillQ[cptenig])

def verif():    #Fonction qui permet de vérifier les réponse aux énigmes
    global cptenig,faux
    for i in range(3):
        if cptenig==i: #Le joueurs effectue la i-ème énigme
            if EntryEnig.get()==EnigVillR[i]:
                cptenig=cptenig+1
                EntryEnig.delete(0,END)
                Enigme()
            else:
                faux=faux+1 #Compteur du nombre d'erreur
                if faux == 5: #Nombre d'erreur limité
                    text(EnigVillF[1])
                    changementdec(1)
                    EntryEnig.config(state = DISABLED)
                    ButtonEnig.config(text="", command=NONE)
        if cptenig==3: #Compteur exprimant le sans faute du joueur
            text(EnigVillF[0])
            inventaire(59)
            changementdec(1)
            EntryEnig.config(state = DISABLED)
            ButtonEnig.config(text="", command=NONE)
```



Quelques fonctions – Fonctions secondaires (1)

```
#Puissance#
def power():    #Fonction qui permet de mettre à jour la puissance du joueur
    global puissance,T
    pt50 = [38,39,59,60]
    pt10 = [45,46,47]
    pt5  = [35,36,57,58]
    for k in range(len(T)): #On parcourt la liste des objets
        if T[k] in pt50:
            puissance = puissance + 50
        if T[k] in pt10:
            puissance = puissance + 10
        if T[k] in pt5:
            puissance = puissance + 5
```



```
##Ramassage#
def ramasser(event):    #Fonction qui permet de ramasser un objet
    global cptmap,M,x,y,son
    objet=[35,36,37,38,39,40,41,42,43,44,45,46,47,52,53,54,55]
    i,j=indices(x,y)
    if M[cptmap][i][j] in objet:
        power()
        inventaire(M[cptmap][i][j])
        changementdec(11)
        ramasse.play()
```

Quelques fonctions – Fonctions secondaires (2)

```
def inventaire(x):          #Fonction qui permet de mettre a jour l'inventaire
    global L,T
    T.append(x)
    L.append(d[x-1])
    affichage(L)

def affichage(L):           #Fonction qui permet l'affichage des objets dans l'inventaire
    for k in range(len(L)):
        can.create_image((120+50*k),30,image=L[k])
```

Aperçu

Quelques fonctions – Fonctions secondaires (3)

Ces fonctions permettent l'optimisation du programme en économisant des lignes de code

```
#Fonction pour configurer les labels et les boutons#
def bouton(text1, text2, command1, command2):
    ButtonCh1.config(text=text1, command=command1)
    ButtonCh1.pack(side=LEFT)
    ButtonCh2.config(text=text2, command=command2)
    ButtonCh2.pack(side=RIGHT)

def text(text):
    LabelDiag.config(text = text)
    LabelDiag.pack()
```

```
#Fonction qui parmet d'appeler les indices i et j
def indices(x,y):
    i=(x-50)//32
    j=(y-100)//32
    return i,j

#Fonction permettant de remplacer une image par une autre dans la matrice
def changementdec(f):
    global cptmap,img,x,y
    i,j=indices(x,y)
    M[cptmap][i][j]=f
    decor(M[cptmap])
    img=can.create_image(x,y,image=fichier_imgd)
```

L’interface Homme-Machine (IHM)

- Plusieurs commandes :
  - <a> : Transition entre les maps
  - <p> : Interaction avec les personnages
  - <z> : Ramasser un objet
  - <Left>, <Up>, <Right>, <Down> : Déplacement du personnage
  - <space> : Transition avec la map de fin de jeu
  - <q> : Affronter le boss (Fin du jeu)

- Interaction avec la fenêtre :
  - Buttons
  - Entry

Problèmes survenus

Problèmes	Solutions
Collisions/Transitions	Création de matrices et de la variable « cptmap »
Taille Canevas/Maps	Redimensionnement
Interface Graphique	Utilisation de Frame, Label Frame, Label, Button, Entry
Dialogues	Un seul compteur par personnage dans chaque fonction
Installation de Pygame	Réitération de la manipulation d'installation

Questions et Screencast