Diego BOUHANICHE

Robin FELLIUS

Alyssia PASTOR

2016/2017

Dossier ISN :

Programmation d'un jeu de rôle textuel en Python

Sommaire

1. Introduction

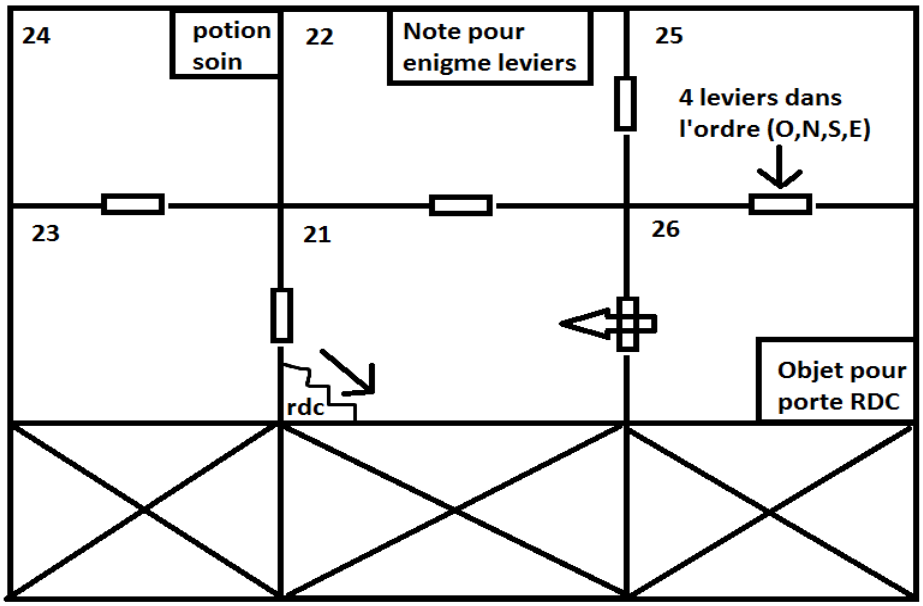
Nous avons choisi de réaliser un jeu de rôle textuel grâce à Python. Dans le jeu, vous incarnez un personnage dont le but est de réussir à s'enfuir du manoir où il s'est réveillé sans aucun souvenir, durant son évasion, votre personnage sera confronté à des ennemis. Nous avons créé un scénario et un univers de jeu : le manoir. Pour répartir le travail nous avons décidé de coder un étage chacun soit entre 6 et 9 salles par personne dont le passage d'une salle à l'autre et les épreuves liés aux salles de notre étage. Nous avons importé une bibliothèque, « import os », cette bibliothèque permet d'effacer les lignes précédentes sur la console de commande lorsque l'on joue et ainsi rendre le jeu plus clair. Robin s'est occupé de créer un système de combat et les salles du premier étage, Diego un système fonctionnel d'inventaire et les salles du rez-de-chaussée alors que je me suis occupé de coder le sous-sol avec ses salles et son énigme.

2. Se déplacer dans le jeu

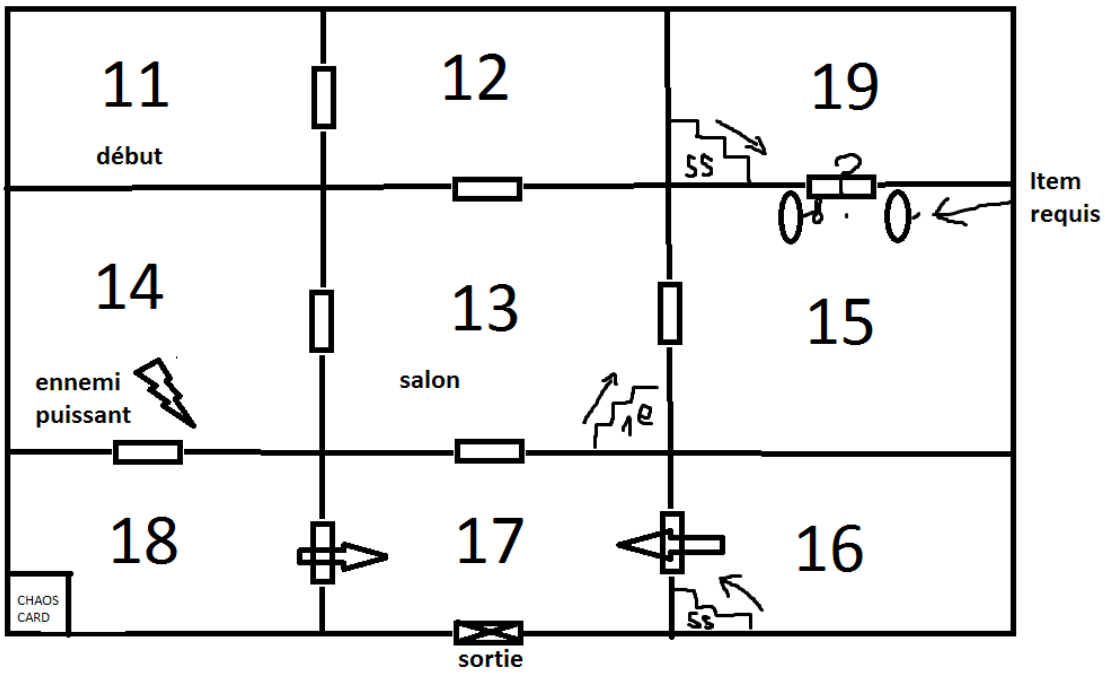
Nous avons numéroté chaque salle du manoir pour ensuite créer une variable « salle » dont la valeur correspond au numéro de la salle dans laquelle le joueur se trouve. Ce numéro permet au programme d'afficher le contenu relatif à la salle où se trouve le joueur.

2.1. Le plan du jeu

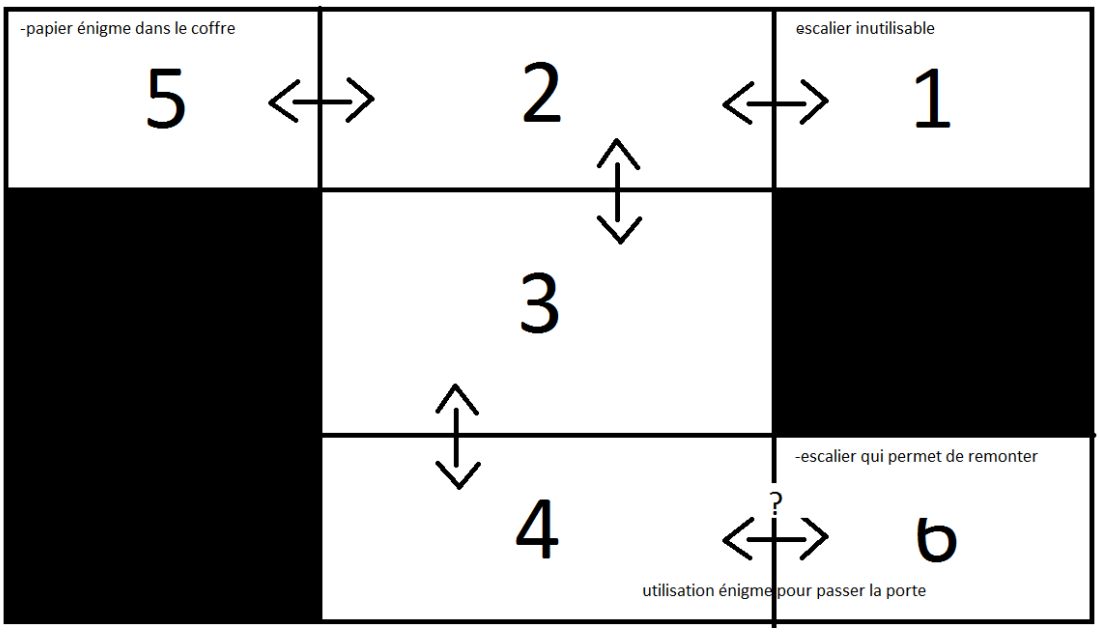
2.1.1. Le plan du premier étage



2.1.2. *Le plan du rez-de-chaussée :*



2.1.3. *Le plan du sous-sol :*



2.2. Le fonctionnement des déplacements :

Pour se déplacer dans le jeu, nous avons initialisé 2 variables, la variable salle, en lui donnant le numéro de salle ou le personnage commence son aventure et la variable fin du jeu.

L'initialisation des variables :

```
findujeu=0

salle=1
```

A la suite nous avons créé une boucle « while fin du jeu est différent de 1 alors donner les instructions possible à accomplir dans la salle :

```
while findujeu!=1:
```

Nous avons créé des conditions avec les valeurs des salles dans lesquelles nous pouvons aller. Cela nous a permis d'intégrer aux salles toutes les actions qu'il est possibles d'effectuer.

```
#salle1----------------------------------------------------------------------------------------------------------------------------------------------------

  if salle==1:

    changersalle=0
```

Pour chaque salle où le joueur arrive, on afficher une phrase qui informe le joueur sur son environnement.

```
    print ("Vous arrivez dans une salle dépourvue de lumière, un grand bruit vous surprend ! L'escalier vient de s'affaisser, il vous faudra trouver un autre moyen de remonter!")
```

Nous avons créé une boucle « while » à l'intérieur de chaque boucle « if salle==… » afin que le joueur puisse agir dans la salle jusqu'à ce qu'il décide de se déplacer.

```
    while changersalle==0:
```

La variable « action » permet au joueur d'interagir avec le jeu et de choisir l'action qu'il veut accomplir.

```
      action=input("Que voulez vous faire ? ")
```

Nous avons ajouté des commentaires grâce à des « # » dans notre programme pour augmenter la lisibilité de ce dernier et ainsi faciliter la mise en commun et les corrections.

```
#les différentes actions :


#Les actions non valides.....................
```

Ici nous avons listé les actions valides afin de faire apparaître un rappel si l'action écrite par le joueur n'est pas valide.

```
        if action!="bouger" and action!="Bouger" and action!="regarder" and action!="Regarder" and action!="utiliser" and action!="Utiliser" and action!="prendre" and action!="Prendre": #rappels en cas d'erreurs

            print("Veuillez entrer une action valide. (Rappel: 'Regarder', 'Bouger', 'Utiliser','Prendre')")
```

Cette partie du programme est une réponse à l'action que le joueur a choisi. S'il a choisi de regarder, un message va s'afficher pour lui annoncer ce qui se trouve dans la pièce.

```
#action regarder et utiliser.....................

        if action=="regarder":

            print ("Il n'y a rien dans cette salle sauf ce vieil escalier qui s'est écroulé")
```

La boucle « if » correspond à l'action de déplacement, à l'intérieur de cette boucle la variable « action » demande à l'utilisateur quelle direction il veut emprunter. Selon son choix une boucle de direction sera utilisé et il pourra changer de salle ou si la direction choisit n'est pas possible une phrase s'affichera pour lui rappeler les directions où il peut aller.

```
#Déplacement.....................................

        if action=="bouger" or action=="Bouger":

            action=input("il n'y a qu'une seule porte qui mène à l'ouest du manoir dans cette pièce, où  voulez-vous aller ? ('Nord', 'Sud', 'Est', 'Ouest')")

            if action!="Ouest" and action!="ouest":

                print("Vous ne pouvez aller qu'à l'Ouest")
```

Ici la direction ouest est possible, la variable salle prendra donc la valeur de la salle situé à gauche et la variable changersalle sera égale à 1 ainsi nous pourrons changer de salle et commencer la boucle « if » suivante correspondant à la salle dans laquelle l'utilisateur à choisit d'aller.

```
            if action=="Ouest" or action=="ouest":

                print("vous empruntez la porte situé à l'ouest")

                salle=2

                changersalle=1
```

os.system est un module qui permet d'effacer la console de commande de python afin de rendre le jeu plus clair. Nous avons décidé de la placer à chaque changement de salle pour rendre l'interface plus clair pour le joueur.

```
                os.system('cls')
```

3. Le système de combat

Comme bon nombre de jeux d'aventures, le notre propose au joueur un système de combat. Il se présente sous la forme d'un duel avec un adversaire dont les caractéristiques augmentent au fur et à mesure de l'aventure. Le but est simple : faire tomber les points de vie de l'adversaire à 0 sans que ceux du joueur ne tombent à 0 (sans quoi le combat est perdu). Le combat se déroule au tour par tour (l'adversaire et le joueurs agissent à tour de rôle) et se termine lorsque l'un des deux n'a plus de point de vie ou si le joueur fuit le combat. Pour intégrer le combat à notre programme, on a utilisé une fonction que l'un utilise à chaque fois qu'un combat a lieu. La fonction débute comme cela :


```
def combat(attaque, vie, defense, vieennemi, attaqueennemi, defenseennemi, potion, carte):
    endurance=4
    fincombat=0
    n=0
    fuite=0
    etat=0
    etatennemi=0
    global potion1
    global carte1
    global vie1
    global issue
    carte1=carte
    potion1=potion
    issue=0
    vie1=vie
    while (fincombat<1):
        print ("  ")
        print ("Vos caractéristiques :  Attaque :", attaque, "      Vie : ", vie1, "        Défense : ", defense,"       Endurance : ", endurance)
        print ("  ")
        print ("Caractéristiques de votre adversaire :   Attaque :", attaqueennemi, "      Vie : ", vieennemi, "        Défense : ", defenseennemi)
        print ("  ")
        dé_n(10)
        if résultatdé<=6:
            actionennemi=1
        else :
            actionennemi=2
        if etat>0:
            actionennemi=1
        if etatennemi>=1:
            print("Votre adversaire est destabilisé et ne peux ni attaquer ni se défendre")
        if etatennemi==0:
            if actionennemi==1:
                print ("Votre adversaire se prépare à vous attaquer !")
                dé_n(100)
                if résultatdé>=60+attaqueennemi*5:
                    actionennemi=4
            if actionennemi==2:
                print ("Votre adversaire se prépare à défendre")
                dé_n(100)
                if résultatdé>=60+attaqueennemi*5:
                    actionennemi=5
        if etat==0:
            if n>0:
                print ("Veuillez choisir une action valide")
            action=input("Voulez vous attaquer, défendre, fuir, vous reposer ou utiliser un objet de votre inventaire ?")
            os.system('cls')
            if action=="utiliser" or action=="Utiliser":
                print("Vous possedez ", potion1," potions de soin, ")
                if carte1==1:
```

Comme bon nombre de jeux d'aventures, le notre propose au joueur un système de combat. Il se présente sous la forme d'un duel avec un adversaire dont les caractéristiques augmentent au fur et à mesure de l'aventure. Le but est simple : faire tomber les points de vie de l'adversaire à 0 sans que ceux du joueur ne tombent à 0 (sans quoi le combat est perdu). Le combat se déroule au tour par tour (l'adversaire et le joueurs agissent à tour de rôle) et se termine lorsque l'un des deux n'a plus de point de vie ou si le joueur fuit le combat. Pour intégrer le combat à notre programme, on a utilisé une fonction que l'un utilise à chaque fois qu'un combat a lieu. La fonction débute comme cela :

(image paint nulle)

L'adversaire peut attaquer ou se défendre, le joueur peut attaquer, se défendre, se reposer, utiliser un objet en sa possession (potion ou carte de bannissement).

Le choix d'action de l'adversaire se fait grâce à une autre fonction (la fonction « dé(n) ») qui donne au hasard un chiffre entre 1 et n. Si ce chiffre est inférieur à une certaine valeur, l'adversaire attaque, sinon il se défend.

Le joueur entre dans la console de commande l'action qu'il souhaite réaliser. La réussite ou non d'une action est définie grâce à un dé dont le résultat doit être inférieur à un certain chiffre (pour l'attaque et la défense ce chiffre est influencé par le niveau du joueur en attaque et en défense). Attaquer contre un ennemi qui se défend vous paralyse pour un tour de jeu, vous défendre alors que votre ennemi attaque peut le paralyser pendant un tour également.

4. L'inventaire

Le système d'inventaire permet à tout moment durant la partie d'afficher et de voir les objets que possède le joueur. Il permet également d'utiliser ces objets, à la manière de l'action « utiliser » basique.

Par exemple si le joueur veut utiliser une potion mais qu'il souhaite vérifier combien il en possède :

```
#Inventaire
    if action=="inventaire" or action=="Inventaire":
        print("Vous posséder les objets suivants :")
        if potion>=1:
            print(potion, "potion(s) de soin")
    action=input("Que souhaitez-vous utiliser ? ")


    if action=="potion" or action=="Potion":
        if potion>0:
            vie=viemax
            potion=potion-1
        else : print("Vous n'avez pas de potion à utiliser")
```

Donc si le joueur demande de voir son inventaire, on lui affiche le nombre d'objets qu'il possède.

S'il décide d'utiliser un objet qu'il n'a pas/plus, on lui annonce que cet objet n'est pas présent dans son inventaire.

5. Les énigmes

La majorité des jeux d'aventure proposent des énigmes. Le notre en propose une par étage ainsi qu'une recherche d'indices tout au long de l'aventure pour en comprendre le récit. En effet à la fin du jeu, une dernière question est posée au joueur pour tester sa compréhension de l'histoire. Les autres énigmes mettent aussi les capacités cognitives et/ou la culture du joueur à l'épreuve.

5.1. Le premier étage

Au premier étage l'énigme consiste à l'activation dans le bon ordre de quatre leviers situés aux quatre points cardinaux. Dans la pièce adjacente le joueur peut trouver une note sur laquelle est écrit « Tout commence au PONANT et finit à l'ORIENT, celui du SUD précède celui de l'EST », autrement dit, il faut activer le levier de gauche (le ponant) en premier et celui de droite (l'orient) en dernier, celui du sud avant celui de l'est donc en avant dernier. L'ordre est donc « ouest, nord, sud, est ». En entrant les directions dans le bon ordre, le joueur gagne un point de score et peut aller dans la salle suivante pour continuer l'aventure.

5.2. Le rez-de-chaussée

Une énigme est présente au rez-de-chaussée. Elle consiste à trouver un objet manquant sur une statue. Si on utilise l'action « Regarder », nous obtenons des détails : « une des statues présente un défaut à LA MAIN, comme s'il manquait une partie... ». L'objet en question est un gant trouvable seulement au 1er étage, ce qui permet de relier le rez-de-chaussée à l'étage pour pouvoir continuer l'histoire.

Une fois le gant récupéré à l'étage, il faut l'utiliser dans la salle avec la statue :

```
action=input("Que souhaitez-vous utiliser ? ")
if action!="gant":
    print("Vous n'avez pas cet objet ou l'objet ne va pas sur la statue")
if action=="gant" or action=="Gant" or action=="Gant de mailles" or action=="gant de mailles" or action=="Gant de maille" or action=="gant de maille":
    if gant==0:
        print("Vous n'avez pas cet objet")
    if gant==1:
        print("Vous posez le gant sur la main de la statue, la porte qu'elles gardaient s'ouvre pour vous laissez l'accès à la salle suivant")
        statue=1
```

```
            gant=gant-1

            score=score+1
```

Si le joueur n'utilise pas le gant, on lui dit que l'objet ne va pas sur la statue. S'il tente d'utiliser le gant alors qu'il ne l'a pas ramassé au 1er étage, on lui dit qu'il n'a pas l'objet (donc la porte ne se déverrouille pas).

S'il utilise le gant, qu'il a ramassé au préalable, et donc, dans son inventaire, alors la variable « statue » se met à « 1 », le gant s'enlève de son inventaire (« gant=gant-1 ») et la porte se déverrouille, le joueur peut donc passer à la salle suivante (Salle 19) :

```
 if action=="Nord" or action=="nord":

            if statue==0:

               print("La porte est bloquée.")

            if statue==1:

               salle=19
```

Le score du joueur est également augmenté de 1 pour la résolution de l'énigme.

5.3. Le sous-sol

Au sous-sol il y a une énigme, dans la salle 5 (cf. Plan du sous-sol page 4) on trouve l'énoncé de l'énigme : « Les riches en manquent, les pauvres en ont et si vous le mangez, vous mourrez. ». L'utilisateur n'a pas besoin tout de suite de la réponse, il l'utilise lorsqu'il choisit la direction Est dans la salle 4 l'utilisateur doit écrire la réponse « rien » pour pouvoir passer dans la salle 6. Cette énigme fonctionne comme une condition : si la réponse est juste l'utilisateur peut emprunter la porte sinon il reste dans la même salle

```
if question=="rien" or question=="Rien":

            print("vous avez bien répondu ce n'était pas simple pourtant")

            changersalle=1

            salle=6

            os.system('cls')

         else:

            print("Ce n'est pas la bonne réponse, il vous faut sûrement plus de temps, la question n'est pas
facile")
```

6. AUTRES FONCTIONS

6.1. LA FONCTION CREATION DE PERSONNAGE

La fonction creationdepersonnage() permet de donner un nom à l'aventurier que vous incarnez et répartir ses points de compétence entre les trois caractéristiques du jeu : l'attaque, la défense et les points de vie. Elle n'est appelée qu'une seule fois au début de l'aventure.

6.2. LA FONCTION DE

Elle permet de donner un nombre entier entre 1 et n (n étant le paramètre de la fonction)

(insérer image « fonction dé »)

6.3. LES FONCTIONS D'AIDE

La fonction tutoriel() est appelée une fois au début du jeu et énonce au joueur l'histoire, le but du jeu et la manière de jouer. Elle explique les mécaniques de gameplay (comment gagner un combat par exemple)

Les fonctions aidecombat() et aideexploration() permettent au joueur qui ne sait plus quoi faire de se remémorer les commandes possibles, les mécaniques de combat ainsi que les subtilités du jeu. Pour cela il n'a qu'a entrer « aide » à n'importe quel moment du jeu. Si il est en combat, aidecombat() sera appelée, si il est en phase d'exploration, aideexploration()

6.4. LA FONCTION DE MONTEE DE NIVEAU

Elle est appelée, une fois tous les deux combats réussis et permet au joueur d'augmenter une de ses caractéristiques (qui sont l'attaque, la défense et la vie)