Rapport de PCO

Shufang ZHANG

SEMESTRE 4 | L2 INFORMATIQUE | 07/04/2022

Sommaire

Table des matières

1. Introduction

Dans le cadre de notre deuxième année pour valider l'UE de PCO, nous avons eu pour tâche la réalisation d'un projet informatique en fonction des demandes.

Notre objectif est de réaliser et de développer un Space Invaders en utilisant le langage python. Le jeu comprend une matrice des aliens et un défenseur qui peut tirer des balles. Le joueur contrôle le défenseur pour se déplacer et tirer des balles à travers le clavier.

Dans ce dossier nous allons expliciter certaines parties délicates de notre code et indiqué dans le sujet que nous décrivons la conception.

2. Les responsabilités des classes

On a commencé par créer la fenêtre principale s'ouvre avec une Frame contenant un Canvas et réaliser des fonctions de base (dessiner le Defender et le faire déplacer avec les touches du clavier, dessiner une matrice des aliens et le faire déplacer, etc.) en complétant des 5 classes : Alien, Fleet, Defender, Bullet, Game et SpaceInvaders qui sont déjà donnée. Après on a développé les 5 classes et créé des autres classes pour développer ce jeu (l'affiche ment du score, etc.).

2.1 des fonctions de base

Dans cette partie on développe la version minimale du jeu en utilisant 5 classes

1) Class Alien

- **But** : Cette classe est une classe pour modéliser l'alien qui se déplace et l'explosion.
- **Idée** : chaque alien est un mobile et a un id pour sauvegarder sa coordonnée. On récupère la coordonnée du projectile, et laisse afficher l'image d'explosion à cette coordonnée.
- **Code** :

  On définit la méthode touched_by pour dessiner une explosion (la valeur boom) dessinée pendant quelques millisecondes (quand un Bullet touche un Alien).

```
class Alien(object):
    def __init__(self):
        self.id = None
        self.alive = True
        self.alien = PhotoImage(file="alien.gif")
        self.boom = PhotoImage(file="explosion.gif")

    def install_in(self, canvas, x, y):
        self.id = canvas.create_image(x, y, image=self.alien, tags="alien")
        #self.id = canvas.create_rectangle(x, y,x+73,y+53,fill="white", tags="alien")
        return self.id

    def move_in(self, canvas, alien, dx, dy):
        self.id=alien
        canvas.move(self.id, dx, dy)

    def touched_by(self,canvas,projectile):
        x1,y1,x2,y2=canvas.bbox(projectile) #recuperation du coord du projectile
        boom = canvas.create_image(x1+(x2-x1)/2, y1+(y2-y1)/2, image=self.boom, tags="boom")
        canvas.after(100,canvas.delete,boom)
```

2) Class Fleet

- **But** : cette classe permet de gérer un groupe d'alien qui sont dessinés dans la matrice
- **Idée** : l'idée c'est de regrouper les aliens dans une matrice qui bouge de gauche à gauche à droite et de droite à gauche aussi , lorsqu'elle atteint un des deux bords elle descend en bas.
- **Code** :

  les méthode get_dx, get_dy, set_dx paramétrer et changer coordonnée du groupe d'alien
  On calcule le score obtenu par la méthode get_score.
  On dessine 3 lignes et 6 colonnes des aliens et dans la méthode install_in on appelle la classe Alien pour créer des aliens du groupe.

```
class Fleet(object):
    def __init__(self):
        self.aliens_lines = 3
        self.aliens_columns = 6
        self.aliens_inner_gap = 10
        self.alien_x_delta = 5
        self.alien_y_delta = 15
        self.width = 600    #la  largeur
        self.height = 400   # hauteur du jeu

        fleet_size = self.aliens_lines * self.aliens_columns
        self.aliens_fleet = [None] * fleet_size

    def get_dx(self):
        return self.alien_x_delta
    def get_dy(self):
        return self.alien_y_delta
    def set_dx(self,new_dx):
        self.alien_x_delta=new_dx

    def get_width(self):
        return self.width
    def get_height(self):
        return self.height
    def get_score(self):
        return (self.aliens_lines * self.aliens_columns - len(self.aliens_fleet))*20
    def install_in(self, canvas):
        self.alien=Alien()
        self.canvas=canvas
        self.x=50   #coordonnée du position initial
        self.y=50
        pos=0
```

La fonction du méthode move_in est de contrôler et limiter le déplacement.

```
    for m in range(0,self.aliens_lines):
        for n in range(0,self.aliens_columns):
            self.aliens_fleet[pos] = self.alien.install_in(self.canvas,self.x,self.y)
            pos+=1
            self.x += self.aliens_inner_gap+73  #le width de chauqe alien est 73
        self.x=50
        self.y += self.aliens_inner_gap+53   # le height de chaque  alien est 53

def move_in(self, canvas):
    if len(self.aliens_fleet)!=0:
        all_rect_ids = self.canvas.find_withtag("alien")
        x1,y1,x2,y2 = self.canvas.bbox("alien")

        if x2>=620:      #si toucher la coté droit
            self.set_dx(-self.get_dx())
            dy = self.alien_y_delta
        elif x1<=0:      #si toucher la  coté gauche
            self.set_dx(-self.get_dx())
            dy = self.alien_y_delta
        else:        #mettre le fleet sur centre
            dy=0

        for s in range(0,len(all_rect_ids)):
            self.alien.move_in(self.canvas,all_rect_ids[s],self.alien_x_delta,dy)

def manage_touched_aliens_by(self,canvas,defender):
    self.canvas=canvas
    self.defender=defender

    for m in range(len(self.defender.fired_bullets)):
        cond_sorti=0    #sortir du boucle pour ne pas depasser la taille du liste
        xb1,yb1,xb2,yb2=self.canvas.bbox(self.defender.fired_bullets[m]) #cobtenir les oordonnées des bullets
        for n in range(len(self.aliens_fleet)):
```

.

Dans cette classe , on a une condition si tous les ennemis ont été détruits ce n'est pas la peine d'exécuter l'animation

Si jamais les les aliens atteignent le bas la partie s'arrête et le joueur a perdu

```
    if self.aliens_fleet[n] != None: #existe des aliens vivants
        xa1,ya1,xa2,ya2=self.canvas.bbox(self.aliens_fleet[n])

        #cond pour savoir si bullet a touché l'alien
        if (xb1>=xa1 or xb2>=xa1) and (xb1<=xa2 or xb2<=xa2) and yb1<=ya2 and yb1>=ya1:
            self.alien.touched_by(self.canvas,self.defender.fired_bullets[m])
            canvas.delete(self.defender.fired_bullets[m])      #suprémer du bullet
            canvas.delete(self.aliens_fleet[n])    #suprémer de l'alien
            self.alien.alive = False
            del self.defender.fired_bullets[m]   #delete de valeur du liste
            del self.aliens_fleet[n] #= None
            cond_sorti=1    #pour sortir du 2eme boucle
            break

if(cond_sorti==1):   #arreter la 2eme boucle
    break
```

Si les aliens arrivent au bout de l'écran leur direction s'inverse et ils vont dans le sens opposé

3) Class Defender

- **But :** Dans cette class on dessine un carré nommée Defender qui peut être déplacé de gauche à droite avec les touches flèches gauche et droite du clavier et tirer des bullets.
- **Code** :

D'abord on définit des méthodes get_lx et get_ly pour obtenir coordonnée du bullet et on change de coordonnée du Defender par des méthodes set_lx et

set_ly.

```
def get_id(self):
    return self.id

def get_firedB(self):    #la liste des bullets tirés
    return self.fired_bullets

def get_lx(self):
    return self.lx
def get_ly(self):
    return self.ly
def set_lx(self,newx):
    self.lx=newx
def set_ly(self,newy):
    self.ly=newy
```

On utilise la méthode install_in pour créer un defender à la coordonnée (lx,ly,lx+w,ly+h), les valeurs w et h sont de la width et la height de carré.
La méthode keypress pour contrôler le déplacement du carré avec les touches flèches du clavier.

```
def install_in(self, canvas):
    self.canvas=canvas
    lx = self.get_lx()
    ly = self.get_ly()
    w , h = self.width, self.height
    self.carre = canvas.create_rectangle(lx , ly , lx+w , ly+h , outline="black",fill="white")

def keypress(self, event):
    xcond=self.get_lx()  #pour ne pas depasser le cadre du jeu
    if event.keysym == 'Left':
        if xcond > 0:    #évider la defender ne sort pas du cadre
            self.set_lx(self.get_lx()-self.move_delta)    #changer la valeur du x
            self.move_in(self.canvas,-self.move_delta)

    elif event.keysym == 'Right':
        if xcond+self.move_delta < self.canvas_width:   #évider la defender ne sort pas du cadre
            self.set_lx(self.get_lx()+self.move_delta)    #changer la valeur du x
            self.move_in(self.canvas,self.move_delta)

    elif event.keysym == 'space':    #la possibilité de tirer
        self.id = "defender"
        self.bullet = Bullet(self.id)
        self.bullet.set_x(self.get_lx())    #les coordoonées de chaque bullet change avec celui de defender
        self.bullet.set_y(self.get_ly())
        self.fire(self.canvas)
```

Le Defender ne peut tirer que des rafales de 8 Bullet au maximum. on définit méthode fire pour limiter le nombre de bullets autorisé et ajout des bullets à l'écran
Dans la méthode move_bullet on récupère la coordonnée de bullet et supprimer les bullets qui sont dehors du cadre.

```
def move_in(self , canvas , dx):    # defender deplacer horizontal
    canvas.move(self.carre, dx, 0)

def fire(self, canvas):
    if len(self.get_firedB()) < self.max_fired_bullets:      # pour limiter le nombre de bullets autorisé
        self.bullet.id = self.bullet.install_in(self.canvas)    #ajout des bullets a l'ecran
        self.fired_bullets.append(self.bullet.id)    #ajoute des bullets tirés dans la liste fired_bullets

def move_bullet(self,canvas):
    for i in range(0,len(self.fired_bullets)):
        x1,y1,x2,y2 = self.canvas.bbox(self.fired_bullets[i])
        if y1<0:    #si le bullet dehors du cadre
            canvas.delete(self.fired_bullets[i])      #suprrimer cet bullet
            del self.fired_bullets[i]   #supprimer la valeur de cet bullet (dans la liste)
            break
        else:
            self.bullet.move_in(self.canvas,self.fired_bullets[i])   # le déplacement des bullets
```

4) Class Bullet

- **But** : Cette class est une class pour modéliser le Bullet qui se déplace à partir de la coordonnée du Defender quand on appuie sur la barre d'espace.
- **Idée** : Les coordonnées initiales de Bullet sont à côté de Defender. Donc il faut récupérer la coordonnée du Defender d'abord.
- **Code** :

  Nous définissons les paramètres du bullet : radius, color, speed du déplacement. Et on récupère la coordonnée du Defender en appelant la méthode get_lx et get_ly de la classe Defender.
  On laisse le bullet déplacer avec un speed dans la méthode move_in et on dessine le bullet de radius 10 à (x,y).

```
class Bullet(object):
    def __init__(self, shooter):
        self.radius = 10
        self.color = "red"
        self.speed = 8
        self.id = None
        self.shooter = shooter
        self.x=Defender().get_lx()
        self.y=Defender().get_ly()

    def get_x(self):        #obtenir coordoonée du bullet
        return self.x
    def get_y(self):
        return self.y
    def set_x(self,newx):      #changement de coordoonée du bullet
        self.x=newx
    def set_y(self,newy):
        self.y=newy

    def install_in(self, canvas):
        x=self.get_x() + 5
        y=self.get_y() - 10
        r=self.radius
        self.id = canvas.create_oval(x,y,x+r,y+r,fill="red")
        return self.id

    def move_in(self,canvas,ball): #déplacement du bullet
        self.id=ball
        canvas.move(self.id, 0, -self.speed)
```

5) Class SpaceInvaders

- **But** : La fenêtre principale s'ouvre avec une Frame contenant un Canvas
- **Idée** : Utilisez la méthode bind de Frame pour lier la méthode play à un événement, qui consiste à cliquer sur le clavier.
- **Code** :

```
class SpaceInvaders(object):
    def __init__(self):
        self.root = tk.Tk()
        self.root.title("Space Invaders")
        self.frame = tk.Frame(self.root)
        self.frame.pack(side="top", fill="both")
        self.game = Game(self.frame)

    def play(self):
        self.game.start_animation()
        self.root.bind("<Key>", self.game.defender.keypress)
        self.root.mainloop()


SpaceInvaders().play()
```

2.2 développement de projet

Pour améliorer le jeu et réaliser la fonction de sauvegarde des scores, on a créé 2 classes : score et resultat .

- **But** : une gestion des joueurs avec la persistance pour l'inscription de joueurs et l'enregistrement des scores dans un fichier.
- **Idée** : cette parti il faut créer un fichier qui peut sauvegarder les score des joueurs et les informations des joueures inscit. Et afficher la liste des scores sauvegardés et des noms des joueurs.
- **Code** :

  On définit une méthode toFile pour sauvegarder des nouveaux informations dans le fichier A, on définit une autre méthode fromFile pour lire les informations du fichier A.

  Dans cette partie on voulait de réaliser afficher des listes d'inscriptions et scores, et mettre à jour les scores dans le fichier après les joueurs termine ses parties. Mais on arrive pas.

```
class score(object):
    def __init__(self, nom_user, nb_score):
        self.nom_user=nom_user
        self.nb_score=nb_score

    def toFile(self, A):
        f = open(A,"w")
        json.dump(self.__dict__,f)# stocker
        f.close()


    def fromFile(cls, A):
        f = open(A,"r") # ouvrir le fichier A
        d = json.load(f) #Chargement de fichier A
        snew=score(d["nom_user"],d["nb_score"])
        f.close()
        return snew

    def __str__(self):
         return str(self.nom_user) + ',' +str(self.nb_score)
```

```
class resultat(object):
    def __init__(self):
        self.lesscores=[]
    def ajout(self, score):
        self.lesscores.append(score)
    def __str__(self):
        chaine=str(self.lesscores[0])
        for e in self.lesscores[1:]:
            chaine=chaine+ "," + str(e)
        return chaine


    def fromFile(cls,A):
        f = open(A,"r")
        #chargement
        tmp = json.load(f)

        liste = []
        for d in tmp:
            #créer un livre
            s=score(d["nom_joueur"],d["nb_point"])
            #l'ajouter dans la liste
            liste.append(s)
        res=resultat()
        res.lesscores=liste
        f.close();
        return res

    def toFile(self,A):
        f = open(A,"w")
        tmp = []
        for s in self.lesscores:
        #créer un dictionnaire
            d = {}
            d["nom_joueur"] = s.nom_joueur
            d["nb_point"] = s.nb_point
            tmp.append(d)
        json.dump(tmp,f)
        f.close();
```

2.3 fonction principal

1) Class game

- **But :** Appel des class et des méthodes précédent pour réaliser des fonction et du jeu, et afficher le score obtenue à l'écran.
- **Idée :**
- **Code :**

  appeler des class defender , fleet et bullet pour les afficher dans la fenêtre et paramétrer la width et height de canvas etc.

```
class Game(object):
    def __init__(self, frame):
        self.frame = frame
        self.fleet = Fleet()
        self.defender = Defender()
        self.canvas_height = self.fleet.get_height()
        self.canvas_width = self.fleet.get_width()
        self.canvas = tk.Canvas(self.frame,width=self.canvas_width,height=self.canvas_height,bg='black')
        self.canvas.pack(padx=5,pady=5)
        self.defender.install_in(self.canvas)
        self.fleet.install_in(self.canvas)

    def get_canvas_height(self):
        return self.canvas_height
    def get_canvas_width(self):
        return self.canvas_width


    def start_animation(self):
        self.animation()
```

Définir un score initial, calculer un nouveau score après chaque coup Alien et le changer et afficher à l'écran.
la méthode animation: cas de perte et cas de gagner et cas de normale
cas de normale: il reste encore des alien vivant dans le fleet, le defender peut tirer des bullets sur des alien
cas de gagner : ne plus d'alien existe
perte : la coordonnée (valeur y2) d'alien est en dessus du defender

```
    def animation(self):
        self.score = StringVar()  #score actuel
        label = Label(self.canvas, textvariable = self.score)
        self.score.set("score : " + str(self.fleet.get_score()))
        self.canvas.create_window(40, 50, window = label)
        if len(self.fleet.aliens_fleet)!=0:   #quand la liste de fleet non vide
            x1,y1,x2,y2 = self.canvas.bbox("alien")
            if y2<380: #en dessus du defender
                self.move_bullets()
                self.move_aliens_fleet()
                self.fleet.manage_touched_aliens_by(self.canvas,self.defender)
                duree = 100    #le temps entre chaque movement
                self.canvas.after(duree,self.animation)
            else: #cas de PERTE
                self.canvas.create_text(self.canvas_width//2,self.canvas_height//2,fill="white",font="Times 20 italic bold",text="Game Over, L
        else: #cas de  GAGNER: quand la liste vide
            self.canvas.create_text(self.canvas_width//2,self.canvas_height//2,fill="white",font="Times 20 italic bold",text="You WON")


    def move_bullets(self):
        if self.defender.get_id() == "defender":
            self.defender.move_bullet(self.canvas)

    def move_aliens_fleet(self):
        self.fleet.move_in(self.canvas)
```

3. Bilan

Pour ce projet, nous avions comme but de réaliser un jeu avec des instructions définies.
On avait deux parties , une qui consiste à faire développer les classes données et l'autre qui est une version améliorée du jeu , les exigences demandées dans la première partie ont été satisfaites, nous avons bien pris soin de suivre ce qu'on nous demandé de faire , par contre nous n'avons pas pu finir la deuxième partie à cause du temps , on a le score du joueur qui est calculé et qui s'affiche en cours de partie et on a fait aussi une fonction qui nous permet de noter dans un fichier le score et l'utilisateur mais ça ne marche pas.
Nous n'avons pas réellement eu de problèmes particuliers sauf le délai temps qui

ne nous a pas permis de finir.
Nous pouvons suite à ce projet, ajouter quelques modifications afin d'améliorer le jeu. Par exemple,ajouter des niveaux c'est à dire si le joueur réussit le niveau 1 il passe au niveau 2 ainsi de suite et au fur à mesure , les aliens et le defender augmentent de vitesse afin de rendre le jeu plus difficile et on propose l'ajout des aliens à plusieurs vies et du son .

4. Conclusion

Dans ce rapport nous avons expliqué, d'une manière générale, les étapes de développement du Space Invaders et nous avons défini les différentes fonctions qu'on a utilisées pour réaliser ce projet.

Lors du développement, nous avons pu y mettre nos connaissances et ce qu'on a assimilé en cours tout en suivant le cahier de charges , mais malheureusement comme il est indiqué en script , on n'a pas pu finir la dernière et troisième partie.

En définitive, ce projet nous a appris comment programmer en python et ça a été également une occasion pour travailler en binôme.